VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

DEMONSTRACE KRYPTOGRAFICKÝCH PROBLÉMŮ FORMOU INTERAKTIVNÍ VZDĚLÁVACÍ HRY

DEMONSTRATION OF CRYPTOGRAPHIC PROBLEMS THROUGH AN INTERACTIVE EDUCATIONAL GAME

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Anežka Fišarová

AUTHOR

VEDOUCÍ PRÁCE — Ing. Marek Mikulec

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Anežka Fišarová ***ID:*** 230890

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Demonstrace kryptografických problémů formou interaktivní vzdělávací hry

POKYNY PRO VYPRACOVÁNÍ:

Kryptografie jako věda přináší řadu teoretických problémů a jejich řešení, která dnes naprosto běžně využíváme v každodenním životě. Pro běžnou veřejnost je ovšem jejich princip a fungování zcela skryt. Cílem práce je vytvořit interaktivní vzdělávací hru, která zábavnou formou představí vybrané oblasti kryptografie a motivuje uživatele k vyřešení kryptografických (případně steganografických) hádanek a šifer pomocí kryptoanalýzy. Při implementaci budou využity a rozšířeny teoretické a praktické znalosti studenta v oblasti kryptologie a bude provedena praktická realizace hry ve vybraném programovacím jazyce. Hra umožní zábavnou vzdělávací formou rozšířit povědomí veřejnosti o problematice a fungování kryptografie a může sloužit k interaktivní prezentaci fakulty například na dni otevřených dveří.

V rámci bakalářské práce student vhodně zvolí konkrétní kryptografické problémy, které teoreticky popíše a ve zvoleném programovacím jazyce s pomocí technologií zvolených v rámci semestrální práce realizuje. Výsledkem práce bude celistvá interaktivní vzdělávací hra. Student rovněž zohlední praktické UX požadavky pro přívětivá uživatelská rozhraní.

DOPORUČENÁ LITERATURA:

[1] WU, Bian a Alf Inge WANG, 2012. A Guideline for Game Development-Based Learning: A Literature Review. Int. J. Comput. Games Technol. [online]. 2012. ISSN 1687-7047. Dostupné z: doi:10.1155/2012/103710

[2] KATZ, Jonathan a Yehuda LINDELL, 2020. Introduction to modern cryptography. B.m.: CRC press. ISBN 1-351-13301-2.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Marek Mikulec

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce je zaměřená na návrh a realizaci kryptografické interaktivní vzdělávací hry. Věnuje se různým kryptografickým šifrám vybraných tak, aby i člověka mimo obor mohly seznámit s danou problematikou názornou a zábavnou formou. Jako programovací jazyk byla použita Java a JavaFX. Práce se sestává ze čtyř částí, z nichž první dvě jsou orientované na teoretickou část kryptografie a druhé dvě na praktické využití a vývoj v kódu. Jako motivace pro hráče bylo využito bodové ohodnocení za správně vyluštěné šifry. Dále práce obsahuje volbu obtížnosti, která má vliv na to, jaké šifry budou hráči prezentovány. Na závěr práce autor reflektuje nad výsledky a zkoumá možnosti dalšího vylepšení uvedených metod.

KLÍČOVÁ SLOVA

ASCII kódování, Atbash šifra, Bifid šifra, Caesarova šifra, GUI, Homofonní šifry, Java, JavaFX, Klasická kryptografie, Kryptoanalýza, Kryptografie, Kódování, Monoalfabetické šifry, Morseova abeceda, Netbeans, počítačová hra, Polyalfabetické šifry, Polybius šifra, Polygrafické šifry, Scene Builder, Substituční šifry, Transpoziční šifry, Vigenèrova šifra

ABSTRACT

The bachelor thesis is focused on the design and implementation of a cryptographic interactive educational game. It focuses on various cryptographic ciphers specifically selected so that people outside the field can be introduced to the subject in a demonstrative and entertaining way. Java and JavaFX were used as the programming language. The thesis consists of four parts, the first two are oriented towards the theoretical part of cryptography and the second two are more focused on the practical application and code development. As motivation for the players, a score for correctly cracked ciphers was used. Furthermore, the work includes a choice of difficulty, which influences which ciphers will be presented to the players. At the end of the thesis, the author reflects on the results and explores ways to further improve the methods presented.

KEYWORDS

ASCII Encoding, Atbash Cipher, Bifid Cipher, Caesar Cipher, GUI, Homophonic Ciphers, Java, JavaFX, Classical Cryptography, Computer Game, Cryptanalysis, Cryptography, Coding, Monoalphabetic Ciphers, Morse Code, Netbeans, Polyalphabetic Ciphers, Polybius Cipher, Polygraphic Ciphers, Scene Builder, Substitution Ciphers, Transposition Ciphers, Vigenère Cipher

FIŠAROVÁ, Anežka. *Demonstrace kryptografických problémů formou interaktivní vzdělávací hry*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 56 s. Bakalářská práce. Vedoucí práce: Ing. Marek Mikulec

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Anežka Fišarová
VUT ID autora:	230890
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Demonstrace kryptografických problémů formou interaktivní vzdělávací hry

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu Ing. Markovi Mikulcovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Úvod **11**

1 Základy Kryptologie **12**
1.1 Základní pojmy . . . 12
1.2 Kryptologie . . . 12
1.2.1 Kryptografie . . . 13
1.2.2 Kryptoanalýza . . . 13
1.2.3 Steganografie . . . 14

2 Šifrovací systémy **15**
2.1 Moderní kryptografie . . . 15
2.1.1 Symetrické šifrování . . . 15
2.1.2 Asymetrické šifrování . . . 16
2.2 Klasická kryptografie . . . 16
2.2.1 Kódování . . . 17
2.2.2 Monoalfabetické šifry . . . 18
2.2.3 Polyalfabetické šifry . . . 19
2.2.4 Polygrafické šifry . . . 21
2.2.5 Homofonní šifry . . . 22

3 Využité technologie **24**
3.1 Programovací jazyk . . . 24
3.1.1 Java . . . 24
3.1.2 JavaFX . . . 25
3.2 Vývojové prostředí . . . 26
3.2.1 Scene Builder . . . 26
3.2.2 NetBeans . . . 26

4 Výsledky studentské práce **28**
4.1 Návrh uživatelského rozhraní . . . 28
4.1.1 Návrh menu . . . 28
4.1.2 Návrh herního prostředí . . . 29
4.2 Kódové řešení programu . . . 30
4.2.1 Třída CryptGameMain . . . 30
4.2.2 Třída MenuController . . . 31
4.2.3 Třída GameController . . . 32
4.2.4 Třída Resource . . . 35
4.2.5 Abstraktní třída pro šifry . . . 35

4.2.6 Implementace šifry PasswordCrack 36
4.2.7 Implementace Morseovy šifry 38
4.2.8 Implementace Caesarovy šifry 39
4.2.9 Implementace šifry Atbash 41
4.2.10 Implementace šifry Polybius 42
4.2.11 Implementace Vigenèrovy šifry 44
4.2.12 Implementace šifry Bifid 47
4.2.13 Implementace herního skóre 48

Závěr **51**

Literatura **52**

Seznam symbolů a zkratek **55**

A Obsah elektronické přílohy **56**

Seznam obrázků

1.1 Komunikace v kryptografickém systému 12
2.1 Dělení šifrovacích systémů 15
2.2 Přehled znaků Braillovy abecedy 17
2.3 Přehled znaků Morseovy abecedy 18
2.4 Tabulka pro Vigenèrovu šifru 20
2.5 Abecední čtverec pro klíč „HESLO“ 22
2.6 Tabulka pro kódování v Polybius a Bifid šifře 22
2.7 Symbolové zastoupení písmen abecedy v Pigpen šifře 23
3.1 Vývojové prostředí NetBeans 27
4.1 Návrh uživatelského rozhraní pro menu 29
4.2 Identita a název volané funkce po stisknutí tlačítka 29
4.3 Návrh uživatelského rozhraní pro hru 30
4.4 Snímek z šifry PasswordCrack 38
4.5 Snímek z Morseovy šifry Morse 39
4.6 Snímek z Caesarovy šifry 41
4.7 Snímek z šifry Atbash 42
4.8 Snímek z šifry Polybius 44
4.9 Snímek z Vigenèrovy šifry 46
4.10 Snímek z šifry Bifid 48
4.11 Snímek z tabulky skóre 50

Seznam výpisů

4.1 Ukázka třídy CryptGameMain 30
4.2 Konstruktor třídy MenuController 31
4.3 Metoda pro výběr obtížnosti 31
4.4 Metoda initialize 32
4.5 Metody pro přesouvání mezi šiframi 33
4.6 Metoda pro kontrolu odpovědi 34
4.7 Metoda pro načítání slov ze souboru 35
4.8 Ukázka abstraktní třídy pro šifry 36
4.9 Ukázka funkce pro kontrolu shodných znaků slov 37
4.10 Ukázka funkce pro zašifrování slova 38
4.11 Ukázka funkce pro generování náhodného celého čísla 39
4.12 Ukázka funkce pro posunutí znaků v abecedě 40
4.13 Ukázka funkce užívající obrácené abecedy k zašifrování hledaného slova 41
4.14 Ukázka funkce generující mřížku abecedy 43
4.15 Ukázka funkce pro zašifrování dle mřížky 43
4.16 Ukázka funkce generující mřížku abecedy 45
4.17 Ukázka funkce pro šifrování Vigenèrovy šifry 45
4.18 Ukázka funkce rozdělující pole na sudé a liché a následné šifrování . . 47
4.19 Ukázka funkce pro přidání nového hráče 48
4.20 Ukázka funkce pro výběr aktuálního hráče 49

Úvod

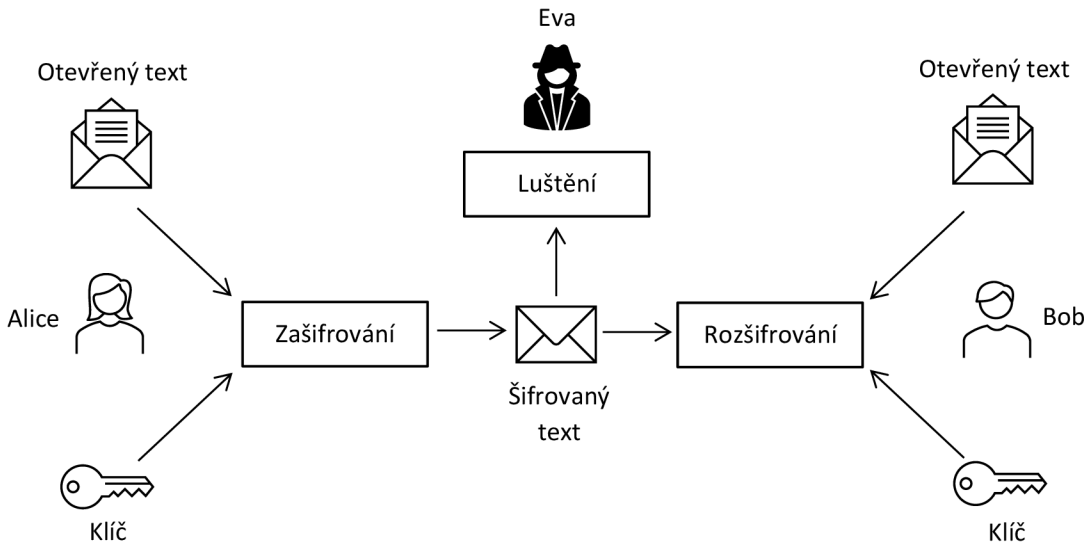
Kryptografie je velmi zajímavý vědní obor, bez kterého by dnešní svět nemohl fungovat. Kryptografie je využívána každý den od šifrování vašeho tajného deníčku až po bezpečnost komunikace na sociálních sítích. I v historii měla značné zastání a byla silně využívána pro tajnou komunikaci v období válek. Člověk měl odjakživa nějaká tajemství, které bylo zapotřebí chránit. Existuje mnoho šifer, z nichž každá se hodí v jiných situacích. Zároveň se šifra někdy jeví jako výzva, hlavně pro jejího autora, který musí vymyslet co nejsložitější šifrový systém. Pro luštitele je to zase naopak výzva šifru rozluštit.

Účelem projektu je seznámit uživatele se základní problematikou kryptografie. Forma počítačové hry je názorná a zábavná, proto v tomto případě bude nejlepší volbou. K tomu bude využit programovací jazyk Java a framework JavaFX, díky kterému bude možné vytvořit uživatelské rozhraní. Bude možné si zvolit obtížnost, která určí složitost šifer. Zároveň se bude každému hráči ukládat skóre za úspěšně vyluštěné šifry.

1 Základy Kryptologie

1.1 Základní pojmy

Strana A (Alice) a strana B (Bob) si předávají soukromě mezi sebou zprávu. Jejich komunikaci však odposlouchává třetí strana E (Eva). Text, který obsahuje zpráva, nazýváme *otevřený text*. Za *šifrový text* označujeme zprávu, která putuje od Alice k Bobovi, a která byla zašifrována (proces tvoření *šifrového textu* z *otevřeného textu*). K zašifrování či rozšifrování zprávy použili Alice a Bob *klíč* (tajnou informaci, kterou si dopředu Alice a Bob vyměnili). Eva tento *klíč* nezná. Tento proces je znázorněn na obrázku 1.1.



Obr. 1.1: Komunikace v kryptografickém systému

Za kryptografický systém se považuje jakýkoliv systém, který lze použít ke změně *otevřeného textu* za účelem učinit ho nesrozumitelným komukoliv jinému až na adresáta.

1.2 Kryptologie

V této kapitole budou uvedeny další základní pojmy týkající se jednotlivých vědních oborů kryptologie. Kryptologie je věda zabývající se utajením obsahu zpráv.
Slovo kryptologie pochází z řečtiny, kde „kryptos“ znamená „skrytý“ a „logos“ znamená „slovo“.
Dělí se na:

- **Kryptografii** (vytváření šifer),
- **Kryptoanalýzu** (lámání šifer),
- **Steganografii** (utajení komunikace).

V historii se převážně používala ve 20. století v době první a poté i během druhé světové války, a to k šifrování válečných zpráv, obchodních korespondencí a podobně důležité komunikace [1].

1.2.1 Kryptografie

Kryptografie, zjednodušeně šifrovaní, se zabývá bezpečností zpráv. Zpráva v tomto případě znamená posloupnost symbolů, ve které je zakódována informace. Symboly v takové zprávě jsou realizovány buďto písmeny (mají textovou podobu) nebo obrazovými body (statický, či pohybující se obraz). Ať už je zpráva realizována jakkoliv, je možné ji převést na posloupnost čísel. Kryptografie se převážně zabývá konstrukcí matematických metod, které nám pomáhají zajistit důvěryhodnost, původ dat (zdroj zprávy), autentizaci subjektů a integritu dat (neporušenost obsahu). Tyto metody nazýváme *šifrovací klíče*. Ti, kteří se zabývají návrhem, použitím a zkoumáním šifrovacích systémů, se nazývají kryptografové [1, 2, 3].

Základní pravidla kryptografie:

- *Klíč* by neměl být použit více než jednou.
- Dbát na dostatečnou délku *klíče*.
- *Klíč* by neměl být jednoduše uhodnutelný.
- V případě použití více *klíčů* by jeden *klíč* neměl být odvoditelný z druhého.
- Kryptografický systém by měl být co nejjednodušší a přehledný.
- Co největší komprese dat [1, 2].

1.2.2 Kryptoanalýza

Kryptoanalýza slouží (naopak od kryptografie) ke zkoumání kryptografických systémů za účelem proniknout do těchto systémů. Obecně je možné kryptoanalýzu definovat jako analýzu odolnosti kryptografických systémů. Zabezpečení kryptografických systémů je možné prolomit využitím slabin použité šifry. Kryptoanalýzou se zabývají kryptoanalytici a jejich cílem je převést *šifrový text* do jeho otevřeného tvaru a získat tak utajovanou informaci [1, 2].

Kryptoanalytické metody:

- Metoda pokus-omyl.
- Frekvenční analýza (zda se jedná o substituční šifru či transpoziční).
- Řešení transpozic v tabulce (postupně přeskupovány bloky v tabulce).

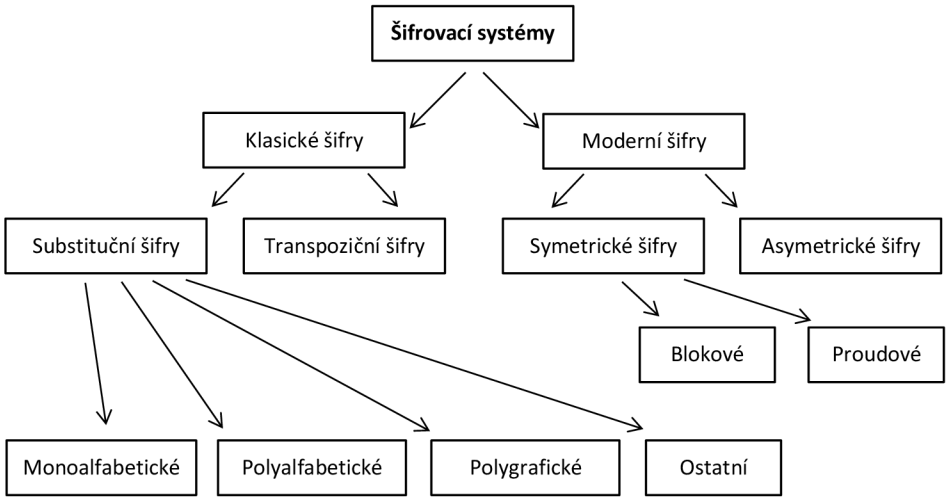
- Řešení polyalfabetických šifer (založeno na výskytu shodných dvojic).
- Hledání *klíče* slovníkovou metodou.
- Brute force attack (útok hrubou silou) a další.

1.2.3 Steganografie

Slovo steganografie pochází z řeckých slov „steganós“ a „gráphein“, což v překladu znamená „skrytý“ a „psát“, čili „skryté psaní“.
Steganografie je tedy věda zabývající se ukrýváním zpráv tak, aby nebyla jejich přítomnost vůbec detekována. Na toto je kladen důraz. Proto, pokud je zpráva odhalena, je považována v podstatě za prolomenou. Aby se zabránilo odhalení, steganografické metody se kombinují s dalšími šifrovacími metodami.
Mezi steganografické pomůcky se například řadí neviditelný inkoust z mléčné šťávy pampelišek nebo citronové šťávy, který se stane po zahřátí viditelným [1, 2].

2 Šifrovací systémy

V kryptografii existuje mnoho různých druhů šifer a proto je seskupujeme do skupin. Tato práce se bude zaobírat především substitučními šiframi, ale pro úplnost bude uvedeno pár dalších skupin šifrovacích systémů. Dělení šifrovacích systémů je znázorněno na obrázku 2.1 převzatého z [4].



Obr. 2.1: Dělení šifrovacích systémů

2.1 Moderní kryptografie

Moderní kryptografie se pohybuje na pomezí matematiky, výpočetní techniky a elektrotechniky. Největší využití má v oblasti ochrany dat ve formách počítačového hesla, digitálního podpisu a také je využívána k autentizaci dat. Moderní kryptografie se dále dělí na symetrické a asymetrické šifrování.

2.1.1 Symetrické šifrování

Za symetrický systém označujeme takový systém, který používá k šifrování i dešifrování stejný *klíč*. Takový *klíč* se poté nazývá *klíčem symetrickým*.

Symetrické šifry se dále dělí na:

1. **Proudové šifry**
   Proudové šifry pracují bit po bitu.
   Příklady proudových šifer:
   - **RC4** (zkratka autorova pseudonymu Rivest Cipher)
     Šifra užívaná pro šifrovaný přenos webových stránek.
     Patent patří firmě RSA.
2. **Blokové šifry**
   Blokové šifry pracují po blocích dat, např. 128 až 256 bitů.
   Příklady blokových šifer:
   - **DES** (Data Encryption Standard)
   - **AES** (Advanced Encryption Standard)

2.1.2 Asymetrické šifrování

V asymetrickém systému se používají *klíče* dva. První je *veřejný klíč* určený k zašifrování *otevřeného textu* a druhý je *soukromý klíč*, který naopak slouží k dešifrování *šifrového textu*.

- **RSA** (zkratka podle jmen autorů)
  RSA algoritmus byl vytvořen roku 1978, spolupracovali na něm Ron Rivest, Adi Shamir a Len Adleman. Proces algoritmu je pomalejší než u symetrických šifer, a to z toho důvodu, že šifrovací i dešifrovací procesy zahrnují hodně výpočtů s velkými čísly. RSA se nejvíce uplatňuje při tvorbě digitálních podpisů, příležitostně i k šifrování symetrických algoritmů [5, 6].

2.2 Klasická kryptografie

Klasická kryptografie je založena na matematice a zakládá si na výpočetní obtížnosti faktorizace velkých čísel. V klasické kryptografii je holý text zakódován do *šifrového textu*, abychom tyto data mohli přenášet po nezabezpečených komunikačních kanálech. Většina šifer používaných laiky jsou právě šifry klasické. Klasická kryptografie má základní dvě složky, a to substituci a transpozici. Substituční šifry nahrazují písmena nebo skupiny písmen jinými a transpoziční šifry umisťují písmena do jiného pořadí. Kódování je také druhem klasické kryptografie [4, 7].

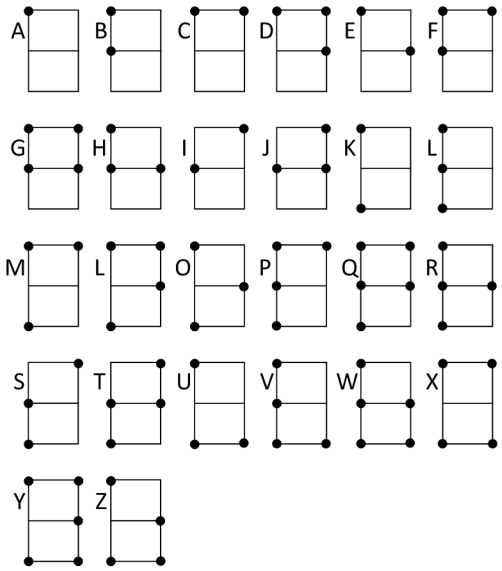
Substituční šifry:

- Monoalfabetické šifry,
- Polyalfabetické šifry,
- Polygrafické šifry a jiné.

2.2.1 Kódování

Kódovaní je činností, při které zašifrujeme informaci pomocí všeobecně známého kódu, který je složen ze znaků. Znakem můžeme rozumět písmeno, číslici či jiné symboly. Klasickým příkladem takových znaků je například Braillovo písmo, kódování ASCII či Morseova abeceda [7].

- **Braillovo písmo**
  Braillovo písmo je způsob psaní textu, který je převážně určený pro nevidomé či slabozraké osoby. Hlavním rozdílem od ostatním druhů kódování je, že se Braillovo písmo specializuje na čtení pomocí hmatu. Každý znak Braillova písma je složen z šesti bodů, které jsou uspořádané do obdélníku v poměru 2x3. V těchto bodech se buďto ocitá vyvýšená tečka nebo prázdné místo. Kombinací teček a prázdných míst bylo vytvořeno 64 jedinečných znaků [8].

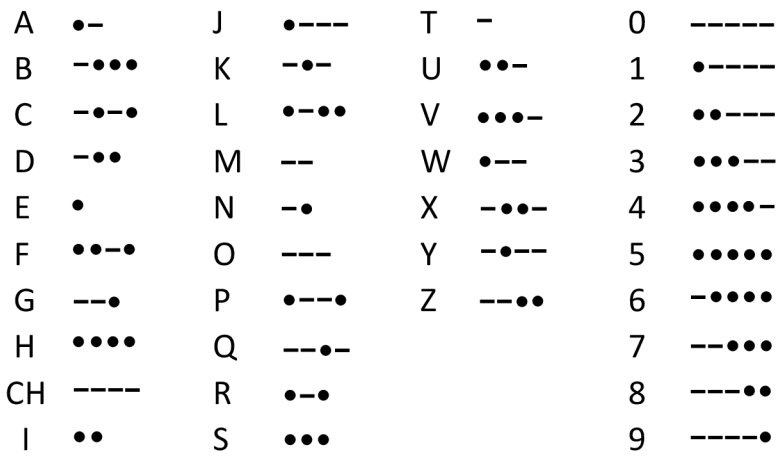


Obr. 2.2: Přehled znaků Braillovy abecedy

- **ASCII kódování**
  V informačních technologiích jsou zpracovány jednotlivé znaky jako čísla. Proto se zavedlo ASCII kódování, které přiřazuje jednotlivým znakům číselné hodnoty. ASCII je zkratkou pro „American Standard Code for Information Interchange“, což se dá volně přeložit jako „americký standardní kód pro výměnu informací“. Každému znaku je přiděleno číslo z intervalu 0-127. Důvodem je, že ASCII je sedmibitový kód. Velikost intervalu se dá dopočítat jako 2^7 [9].

- **Morseova abeceda**
  Ke kódování písmen jsou zde využity krátké a dlouhé signály. Výhoda Mor-

seovy abecedy je v přenositelnosti na značné vzdálenosti a různých možnostech její realizace. Lze s ní komunikovat pomocí světla, zvuku nebo například i impulsů elektrického napětí. Pro zjednodušení se k zapamatování Morseovy abecedy využívají pomocná slova, jejíchž začínající písmeno chcete vyjádřit. Samohlásky, ze kterých je slovo složeno, rozdělíme podle délky na krátké a dlouhé. Krátké samohlásky nahradíme tečkou a dlouhé čárkou [10].



Obr. 2.3: Přehled znaků Morseovy abecedy

2.2.2 Monoalfabetické šifry

Monoalfabetická šifra je jednou z nejjednoduších šifer, jelikož přiřadí každému znaku abecedy jiný znak ze stejné abecedy, dle daného *klíče*. Její největší výhoda spočívá v její jednoduchosti. Nevýhodou monoalfabetické šifry je její zranitelnost vůči frekvenční analýze [11].

Patří sem například:

- **Caesarova šifra**
  Jak název značí, tato šifra byla vytvořena Juliem Caesarem pro vojenskou komunikaci. Princip spočívá v posunutí znaků abecedy o určený počet míst, který je pevně zvolen. *Klíč*, který určuje posunutí, nemá cenu volit větší, než je velikost abecedy. Důvodem je, že pokud zvolíme velikost *klíče* o jedna větší než je velikost abecedy, výsledný posun bude opět jedna [11].

  Příklad:
  Volný text = tajemstvi, klíč = 3
  Abeceda:
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  Posun v abecedě o klíč:

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
Šifrový text = wdmhpvwyl

- **Atbash**
  Atbash je podobný Caesarově šifře s tím rozdílem, že šifrujeme v abecedě od konce. Nahrazujeme tedy první znak abecedy posledním znakem [11].

  Příklad:
  Volný text = tajemstvi
  Abeceda:
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  Abeceda pozadu:
  Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
  Šifrový text = gzqvnhger

2.2.3 Polyalfabetické šifry

V polyalfabetických šifrách se používá víc než jedna šifrová abeceda. Každému znaku je poté možné přidělit víc než jedna náhrada. Typickým příkladem polyalfabetické šifry je Vigenèrova šifra.

Patří sem například:

- **Vigenèrova šifra**
  Jedná se o šifru, která využívá více Caesarových šifer závislých na písmenech *klíče*. Každá z těchto Caesarových šifer má vlastní délku *klíče* (posunu v abecedě). K šifrování se zde používá tzv. tabulka recta, která se skládá z 26 řádků, z nichž každý řádek abecedy se cyklicky posouvá doleva v porovnání s předchozím. Nad touto tabulkou najdeme abecedu pro písmena volného textu a po levé straně zase pro písmena *klíče*. Důležité zde je aby délka *klíče* odpovídala vždy délce volného textu, pokud neodpovídá, opakujeme *klíč*, dokud nedosáhneme stejné délky [12, 13].

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Obr. 2.4: Tabulka pro Vigenèrovu šifru

- **Bifid šifra**

  Bifid šifra je polyalfabetická šifra, která byla vyvinuta ve 20. století. Tato šifra využívá pravidelného čtvercového mřížkového klíče k převodu písmen na čísla a zpět. Princip Bifid šifry spočívá v převodu písmen na souřadnice v mřížce a následném získání nových souřadnic, které jsou převedeny zpět na šifrovaná písmena. Bifid šifra často využívá polybiusovskou mřížku viz obrázek 2.6 pro zobrazení písmen a jejich souřadnic.

  Základním krokem při použití Bifid šifry je převod písmen zprávy na souřadnice ve čtvercové mřížce. Každé písmeno se nahrazuje dvěma čísly, která odpovídají jeho řádku a sloupci v mřížce. Poté jsou tyto čísla zploštěna do jednoho seznamu.

  Následuje druhý krok, ve kterém jsou původní souřadnice zploštěného seznamu rozděleny na dvě poloviny. Tyto poloviny jsou následně použity ke získání nových souřadnic, které představují šifrovaná písmena.

  Ve třetím kroku jsou nové souřadnice převedeny zpět na šifrovaná písmena podle mřížky.

  Mřížka může a nemusí obsahovat po sobě jdoucí písmena abecedy. Pro vyšší bezpečnost je ale lepší zvolit náhodně pořadí [14, 15].

2.2.4 Polygrafické šifry

V polygrafických šifrách jsou písmena nahrazovány skupinami.

Patří sem například:

- **Šifra Playfair**
  V této šifře jde o diagramovou záměnu. Jsou zde aplikována pravidla úpravy a pravidla záměny. Je zde využit abecední čtverec, podle kterého se zašifruje volný text. Jelikož je tento čtverec odvozen z *klíče*, je těžké šifru bez *klíče* rozluštit. Abecední čtverec se vytvoří tak, že do prvního řádku se doplní *klíč*, který tak určí velikost čtverce. Do zbytku se doplní abeceda s výjimkou písmen již obsažených v *klíči* [16, 17].

  Pravidla úpravy:
  1. Celý volný text je zbaven háčků, čárek apod. Pokud obsahuje text písmeno J, je nahrazeno písmenem I.
  2. Písmena volného textu jsou rozdělena do dvojic.
  3. V případě, že je v dvojici shoda písmen, rozdělíme je písmeny X nebo Z.
  4. Jestliže má výsledný text lichý počet, doplníme na konec písmeno Z nebo X.

  Pravidla záměny:
  Dvojice znaků může být v abecedním čtverci buďto ve stejném řádku, sloupci nebo nejčastěji ani v jedné z variant.
  1. Pokud se obě písmena ve dvojici nachází na stejném řádku, je každé z těchto písmen nahrazeno písmenem vpravo od něj. V případě, že vpravo již není písmeno, počítáme s písmenem na začátku téhož řádku.
  2. Pokud se obě písmena ve dvojici nachází ve stejném sloupci, jsou tato písmena nahrazeny písmeny pod nimi. V případě, že pod písmenem již není písmeno, počítáme s prvním písmenem téhož sloupce.
  3. Jestliže písmena z dvojice nespadají do stejného řádku ani sloupce, je každé z těchto písmen nahrazeno písmenem, nacházejícím se v průsečíku jeho řádku a sloupce s písmenem druhým z dvojice. Musí se zachovat pořadí písmen z dvojice.
  4. Jelikož I a J jsou zde identická, může je šifrovatel libovolně zaměňovat.
  5. Posledním krokem je smazání mezer rozdělující jednotlivé dvojice. Pro zmatení je možné vložit interpunkce či náhodné mezery.

H	E	S	L	O
A	B	C	D	F
G	I	K	M	N
P	Q	R	T	U
V	W	X	Y	Z

Obr. 2.5: Abecední čtverec pro klíč „HESLO“

2.2.5 Homofonní šifry

- **Polybius šifra**
  Šifra Polybius je jednoduchá šifra, která byla pojmenována po starořeckém historikovi Polybiovi. Pracuje tak, že převádí jednotlivá písmena abecedy na číselné páry. Tato šifra je založena na čtvercové mřížce, která obsahuje čísla od 1 do 5 na osách x a y. Každé písmeno je poté zakódováno pomocí souřadnic, které odpovídají jeho pozici v mřížce [18, 19].

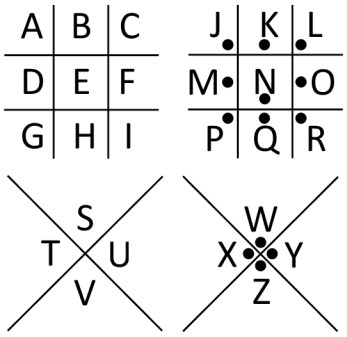
  Princip šifry Polybius je následující:
  1. Vytvoří se čtvercová mřížka s čísly od 1 do 5 na osách x a y.
  2. Každé písmeno abecedy je přiřazeno k souřadnicím ve mřížce. Například A = 11, B = 12, C = 13 atd.
  3. Text, který chceme zašifrovat, je rozdělen do jednotlivých písmen.
  4. Každé písmeno je nahrazeno odpovídajícím číselným párem nebo symbolem podle jeho pozice v mřížce.

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Obr. 2.6: Tabulka pro kódování v Polybius a Bifid šifře

- **Pigpen šifra**
  Je to jednoduchý druh substitučního šifrování, který nahradí každé písmeno

abecedy odpovídajícím symbolem. Pro dešifrování se poté provádí opačný postup, neboli jsou zašifrované symboly opět převedeny zpět na písmena abecedy. Pigpen šifra je využívána spíš pro vizuální efekt než přímo kvůli kryptografické bezpečnosti. Je vhodná pro šifrování krátkých zpráv nebo pro zábavné účely, ale pro silnou šifrovací ochranu se doporučují jiné, pokročilejší algoritmy [20].



Obr. 2.7: Symbolové zastoupení písmen abecedy v Pigpen šifře

3 Využité technologie

V této kapitole budou rozebrány základní technologie, které byly v tomto projektu využity převážně k tvorbě praktické části.

3.1 Programovací jazyk

Programovací jazyk slouží ke komunikaci mezi počítačem a programátorem. Díky programovacímu jazyku může programátor aplikovat řešení na nějaký problém a tento algoritmus poté přenést do počítače. Programovacích jazyků existuje mnoho, z nichž každý je rozdělen podle různých kritérií [21].

Dle míry abstrakce:

- Vyšší programovací jazyky - zápis kódu je přiblížen myšlení člověka,
- Nižší programovací jazyky - strojový kód.

Dle způsobu překladu a spuštění:

- Kompilované programovací jazyky,
- Interpretované programovací jazyky.

Způsob dělení dle překladu a spuštění není zcela absolutní, jelikož mnoho programovacích jazyků spadá pod kompilované i interpretované jazyky (například Java). Také jsou často oba způsoby kombinované dohromady. Například když je zdrojový kód kompilován do mezikódu, a ten poté interpretován [21].

Vyšší programovací jazyky se dále dělí:

- Strukturované (například C a BASIC),
- Objektově orientované (například Java),
- Funkcionální,
- Logické.

3.1.1 Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems a představila ho v roce 1995. Je to jeden z nejpoužívanějších programovacích jazyků na světě. Její velkou výhodou je, že je multiplatformní. Společnost Oracle je současným oficiálním vlastníkem. Implementace Oraclu je rozdělena do dvou různých distribucí: Java Runtime Environment (JRE), která obsahuje části Java SE platformy potřebné pro běh programů v jazyce Java a je určena pro koncové uživatele. Druhou distribucí je Java Development Kit (JDK), který je určen pro

softwarové vývojáře a obsahuje vývojové nástroje jako je Java kompilátor, Javadoc, Jar a debugger.

Veškeré zdrojové kódy jsou volně přístupné, jinak řečeno open source. Největší předností je, že program napsaný v jazyce Java lze spustit na libovolném operačním systému, který má nainstalovanou podporu Java aplikací. Zdrojový kód programu se nejprve překládá do bytového kódu, který se následně přeloží do strojového kódu podle toho, na kterém zařízení je spuštěn.

Další výhodou je silná typová kontrola. Každá proměnná musí mít předem jasně daný datový typ. Kontrola datových typů se provádí už během kompilace kódu. Veškeré chyby způsobené špatnými datovými typy se zobrazí už během kompilace. Programátorovi pak nehrozí, že by takovou chybu přehlédl [22, 23].

Edice Javy, které platí pro JDK 8:

- Java SE (Java Standard Edition) – konzolové, desktopové GUI aplikace,
- Java EE (Java Enterprise Edition) – webové, databázové aplikace,
- Java ME (Java Micro Edition) – vývoj software pro malá zařízení s omezenými prostředky (mobilní telefony s operačním systémem).

K čemu se Java využívá:

- Používá se pro vývoj aplikací pro Android.
- Pomáhá vytvářet podnikový software.
- Existuje široká škála mobilních Java aplikací.
- Užitečná pro vědecké výpočty.
- Používá se pro analýzu velkých dat.
- Programování hardware zařízení.
- Využití pro technologie na straně serveru (např. Apache, JBoss, ...)

3.1.2 JavaFX

JavaFx je sada grafických a mediálních balíčků. Pomáhá vývojářům vytvářet klientské aplikace, které fungují stejně na různých platformách. Vzhled a chování JavaFX lze upravit pomocí kaskádových stylů (CSS). JavaFX má dvě možnosti vývoje, buďto vývojář píše kód samostatně a nebo je možné využít JavaFX Scene Builder [24].

Klíčové vlastnosti JavaFX:

- JavaFX je hnihovnou programovacího jazyka Java.
- Obsahuje vestavěné ovládací prvky uživatelského rozhraní. Jednotlivé komponenty je možné skinovat s využitím standardních webových technologií jako je CSS.
- Poskytuje funkce 3D grafiky.

- Obsahuje webovou komponentu WebView, která umožňuje vkládat webové stránky do aplikace JavaFX.
- Je kompatibilní se stávající aplikací Swing a další.

3.2 Vývojové prostředí

3.2.1 Scene Builder

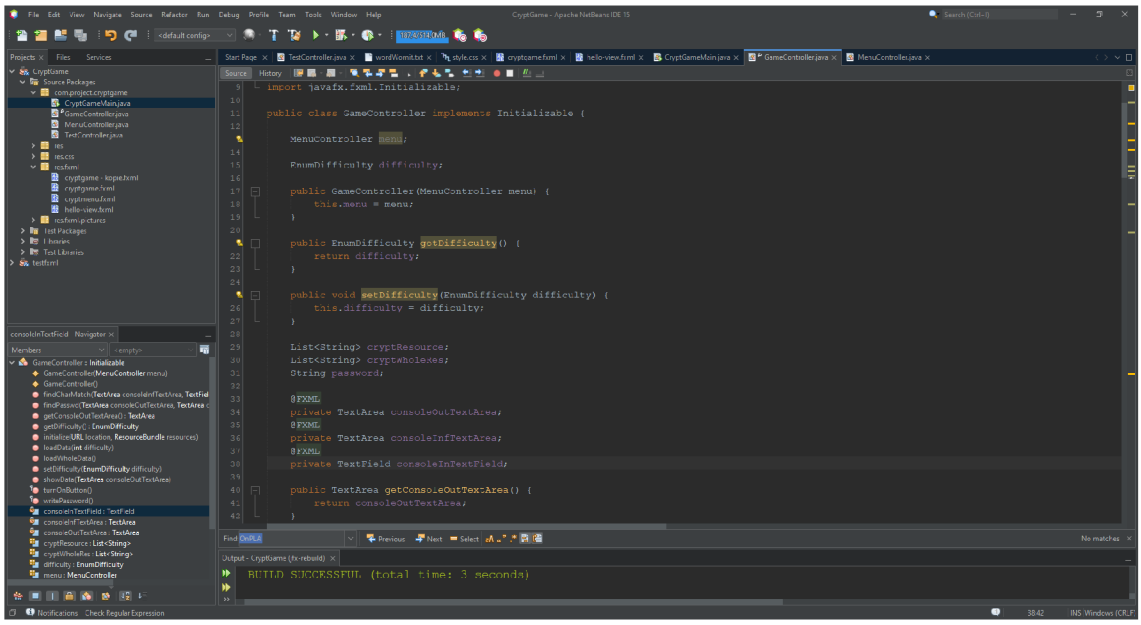
JavaFX Scene Builder je nástroj pro vizuální rozvržení. Uživatelé mají možnost přetahovat komponenty do pracovní oblasti a používat šablony stylů. Na pozadí je automaticky generovaný kód FXML ve kterém je uložené rozvržení. Výsledný FXML soubor lze následně zkombinovat s projektem v jazyce Java a navázat tak uživatelské rozhraní na logiku aplikace. Ukázka rozložení prostředí Scene Builderu je vidět na obrázku 4.1 [25].

3.2.2 NetBeans

NetBeans je open source vývojové prostředí s rozsáhlou uživatelskou základnou a značnou komunitou vývojářů.

NetBeans IDE je určené hlavně pro vývoj aplikací v jazyce Java (Java SE, Java EE, Java ME). Podporuje ale také další programovací jazyky (například C/C++, PHP, HTML5/CSS, Java Script). Celé prostředí je naprogramováno v jazyce Java.

Obsahuje integrované vývojové prostředí, ve zkratce IDE (Integrated Development Enviroment). IDE je navrženo tím způsobem, aby omezilo chyby v kódu a usnadnilo programátorovi jejich opravy. K opravám chyb je například používán Debugger, díky kterému může programátor projít každý řádek svého kódu a zkontrolovat zda funguje jak zamýšlel. NetBeans editor automaticky odsazuje řádky, spojuje slova a závorky, a systematicky zvýrazňuje zdrojový kód. Umožňuje snadno refaktorovat kód, řadou užitečných a výkonných nástrojů, přičemž také poskytuje šablony kódu a kódovací tipy [26].



Obr. 3.1: Vývojové prostředí NetBeans

4 Výsledky studentské práce

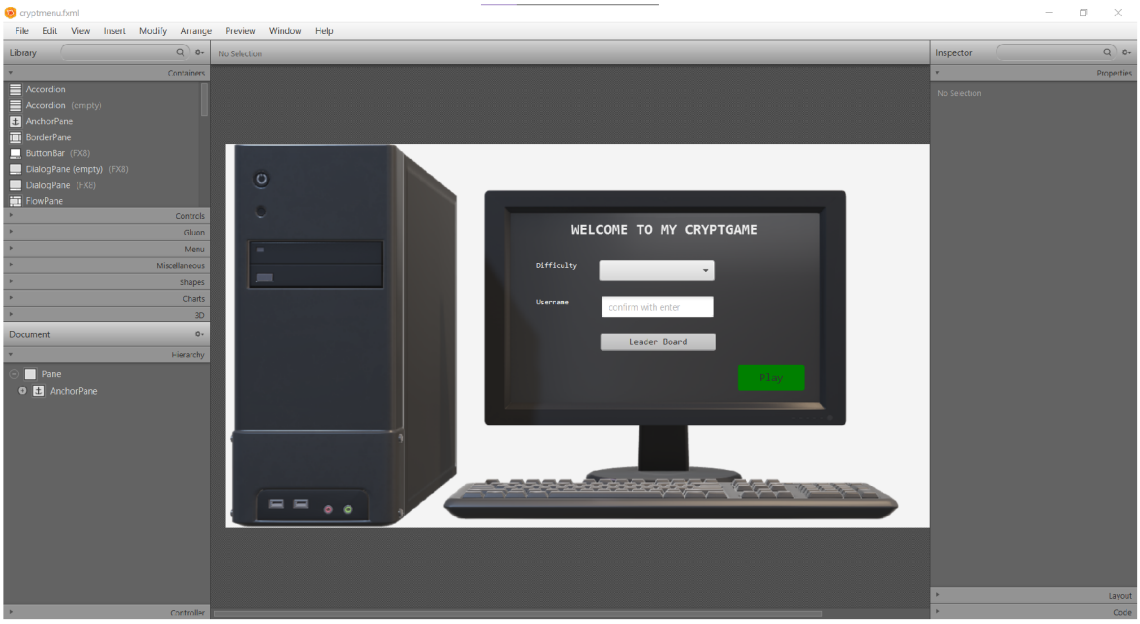
V této kapitole bude rozebrán postup řešení praktické části projektu. Bude zde popsán proces vývoje uživatelského rozhraní a také kódové řešení programu.

4.1 Návrh uživatelského rozhraní

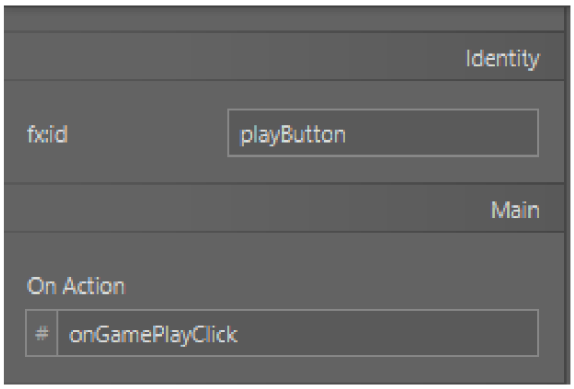
K návrhu uživatelského rozhraní byl využit program Scene Builder, který je založen na frameworku JavaFX. Prvním krokem bylo seznámení se s prostředím a jeho komponenty.

4.1.1 Návrh menu

Nejprve byl vytvořen jednoduchý návrh uživatelského rozhraní pro menu 4.1.V tomto menu byly využity různé komponenty k dosažení požadované funkčnosti. Jako základní kontejnery byly použity `Pane` a `AnchorPane`. Na tyto kontejnery byl umístěn obrázek počítače, který byl získán z volně přístupných 3D modelů v textovém procesoru programu Microsoft Word, součásti balíčku Microsoft Office. Následně byl tento model uložen jako obrázek a použit pro přehlednější zobrazení menu aplikace. Důraz byl kladen především na rozmístění komponent z hlediska funkčnosti aplikace, spíše než na estetiku. Prvním prvkem umístěným v menu bylo tlačítko `Play`, které slouží k otevření nového okna pro samotnou hru. Důležité bylo také vyplnění `id` tohoto tlačítka a názvu funkce, která bude vyvolána po jeho stisknutí (viz obrázek 4.2). Pro uvítání uživatele v aplikaci byl přidán `Label` s textem „WELCOME TO MY CRYPTGAME!“. Následovaly drobné úpravy vzhledu tlačítka. Dále byl přidán `ComboBox`, který umožňuje uživateli vybrat obtížnost hry. Nakonec byly přidány další komponenty, jako například `ChoiceBox` pro výběr šifry, pole pro vyplnění uživatelského jména („Username“) a tlačítko, které slouží k otevření nového okna se seznamem 10 nejlepších hráčů a jejich skóre („Leaderboard“).



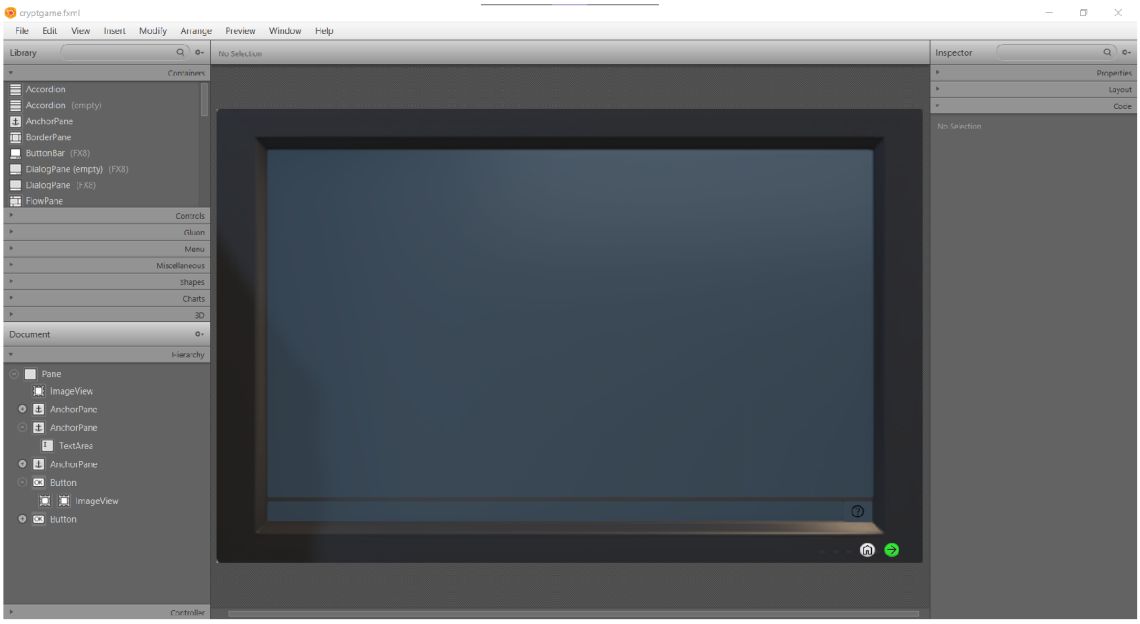
Obr. 4.1: Návrh uživatelského rozhraní pro menu



Obr. 4.2: Identita a název volané funkce po stisknutí tlačítka

4.1.2 Návrh herního prostředí

V tomto návrhu byl klíčovým faktorem vzhled okna, který má evokovat počítačovou konzoli (viz obrázek 4.3). Jako kontejnery zde byly použity `Pane` a `AnchorPane`, stejně jako v návrhu menu. Do `Pane` byl umístěn obrázek monitoru, který byl získán z volně dostupného 3D modelu v textovém procesoru programu Microsoft Word, součásti kancelářského balíčku Microsoft Office. Na tomto obrázku jsou následně umístěny dvě komponenty pro zobrazení textu zadání šifry, konkrétně `TextArea`. Dále je zde také `TextField`, který slouží k získání odpovědi od uživatele.



Obr. 4.3: Návrh uživatelského rozhraní pro hru

4.2 Kódové řešení programu

V této části se zaměříme na kód, který využívá grafického uživatelského rozhraní (GUI) pro správné fungování aplikace. Následující části kódu představují základní myšlenky, postupy a ukázky důležitých částí kódu. Fungování je doprovázeno komentáři v kódu, nad nebo pod ním.

4.2.1 Třída CryptGameMain

Následující kód představuje třídu obsahující hlavní metodu, která se vykoná při spuštění programu. Její hlavní funkcí je inicializace a zobrazení hlavního okna s načteným menu.

Výpis 4.1: Ukázka třídy CryptGameMain

```
1 ...
2     static Scene menuScene;
3     static Stage stage;
4     static Stage menuStage;
5
6     // Metoda pro spuštění nového okna menu
7     @Override
```

```
8       public void start(Stage primaryStage) throws IOException
            {
9           stage = primaryStage;
10          menuStage = stage;
11          FXMLLoader fxmlLoader = new FXMLLoader(CryptGameMain.
                class.getResource("cryptmenu.fxml"));
12          menuScene = new Scene(fxmlLoader.load());
13          primaryStage.setScene(menuScene);
14          primaryStage.show();
15      }
16  ...
```

Používáme zde `fxmlLoader`, který načte fxml soubor, v tomto případě soubor s menu, a otevře ho v novém okně. Důležitým krokem zde bylo vytvoření reference na objekty `Stage` a `Scene`, abychom k nim později mohli přistoupit a vykreslovat scény do jednoho okna.

4.2.2 Třída MenuController

Tento kód představuje třídu `MenuController`, která implementuje rozhraní `Initializable`. Obsahuje metody a události pro ovládání menu aplikace CryptGame. Třída obsahuje různá pole a prvky uživatelského rozhraní, které slouží k výběru obtížnosti, zadání uživatelského jména a spuštění hry. Dále také obsahuje metody pro manipulaci s herní scénou, přidávání hráčů do seznamu a zobrazování žebříčku hráčů.

Výpis 4.2: Konstruktor třídy MenuController

```
1   ...
2       public MenuController() {
3           difficulty = (EnumDifficulty.EASY);
4           game = new GameController(this);
5       }
6   ...
```

Konstruktor je vytvořen bez parametrů a volá se při vytváření instance třídy `MenuController`. V tomto případě i umožňuje propojit mezi sebou třídy `MenuController` a `GameController` za účelem možnosti předání vybrané hodnoty obtížnosti mezi třídami.

Výpis 4.3: Metoda pro výběr obtížnosti

```
1   ...
2       @FXML
3       protected void chooseDifficulty() {
```

```
4           difficulty = (EnumDifficulty) difficultyComboBox.
               getValue();
5           game.setDifficulty(difficulty);
6       }
7   ...
```

Metoda `chooseDifficulty` je označena anotací `@FXML` a je vyvolána jako reakce na událost výběru obtížnosti v poli `difficultyComboBox`. Tato metoda aktualizuje vybranou obtížnost hry na základě získané hodnoty a předá tuto obtížnost do herního objektu `game`.

Výpis 4.4: Metoda initialize

```
1   ...
2       @Override
3       public void initialize(URL location, ResourceBundle
           resources) {
4       difficultyComboBox.getItems().setAll(EnumDifficulty.
           values());
5            difficultyComboBox.setValue(EnumDifficulty.EASY);
6            GameScore.loadScoreFile();
7            GameScore.sortForScore();
8            GameScore.setClearListPlayersonTen();
9       }
10  ...
```

Metoda `initialize` je vyvolána po inicializaci ovládacích prvků ve scéně. v této metodě jsou prováděny různé inicializační úkoly, jako je nastavení dostupných hodnot obtížnosti (EASY, MEDIUM, HARD) v poli `difficultyComboBox`. Obtížnost, neboli `difficulty`, je výčtovým typem, kde EASY = 10, MEDIUM = 20 a HARD = 30. Číselné hodnoty výčtového typu zde značí bodové hodnocení za vyluštění šifry v dané obtížnosti. Poté funkce nastaví výchozí hodnoty v `difficultyComboBox` na EASY, načtení skóre ze souboru, seřazení hráčů podle skóre a omezí seznam hráčů na deset záznamů.

4.2.3 Třída GameController

Třída `GameController` implementuje rozhraní `Initializable` a obsahuje metody a události pro ovládání hry v aplikaci CryptGame. Třída obsahuje různá pole a prvky uživatelského rozhraní, které slouží k zobrazování a ovládání herního prostředí. Vykonávají se zde všechny funkce spojené s fungováním hry.

Funkce třídy jsou následující:

- Přechod na další úroveň hry,
- Přesouvání se mezi šiframi a zpět do hlavního menu,
- Nastavení obtížnosti hry,
- Zobrazování nápovědy k šifrám,
- Ověření odpovědí,
- Přidělování bodového ohodnocení - skóre,
- Zobrazení šifer,
- Inicializace herního prostředí.

Výpis 4.5: Metody pro přesouvání mezi šiframi

```
...
    @FXML
    protected void goToNext() {
        if (level == 3) {        // cyklus mezi šiframi
            level = 0;
        }
        level++;
        System.out.println("Current lvl: " + level);
        setGameObject();
        showCipher(consoleOutTextArea);
    }
    public void setGameObject() {
        this.res = new Resource(difficulty);
        switch (difficulty) {
            case EASY:
                if (level == 1) {
                    this.gameObject = new PasswdCrack(res);
                }
                if (level == 2) {
                    this.gameObject = new MorseCipher(res);
                }
                if (level == 3) {
                    this.gameObject = new CaesarCipher(res);
                }
                break;
            case MEDIUM:
...
```

Funkce `goToNext` slouží k přechodu na další úroveň hry. Pokud je aktuální úroveň rovna 3, nastaví se úroveň na hodnotu 0 (cyklování šifer), poté se úroveň zvýší o jedna, aktualizuje se aktuální herní objekt a zobrazí se nová šifra.

Další funkce `setGameObject` slouží k nastavení herního objektu (šifry) na základě aktuální obtížnosti a úrovně hry. Na začátku se vytvoří instance třídy `Resource` s předanou obtížností. Poté se pomocí přepínače podle aktuální obtížnosti a úrovně nastaví správný herní objekt (`gameObject`). Každá obtížnost a úroveň má svůj specifický herní objekt, který je vytvořen na základě dostupných šifer.

Výpis 4.6: Metoda pro kontrolu odpovědi

```
...
    @FXML
    protected void checkAnswer() {
        int score = GameScore.listPlayers.get(GameScore.
           selectPlayer(menu.getUsernameTextField().getText()
           )).score;
        switch (difficulty) {
            case EASY:
            case MEDIUM:
            case HARD:
                if (level == 1) {
                    if (gameObject.insertAnswer(
                       consoleOutTextArea, consoleInfTextArea
                       , consoleInTextField)) {
                        consoleOutTextArea.setText(gameObject
                           .win());
                        showHintButton.setDisable(true);
                        consoleInTextField.setEditable(false)
                           ;
                        consoleInTextField.setDisable(true);
                        score += difficulty.value;
                    }
                }
                if (level == 2) {
...
```

Funkce `checkAnswer` slouží k ověření odpovědi hráče na aktuální šifru. Nejprve se získá skóre hráče na základě zadaného uživatelského jména. Poté se v závislosti na obtížnosti a úrovni hry provádí kontrola odpovědi. Pokud je aktuální úroveň 1, 2 nebo 3 a hráč zadal správnou odpověď, dojde k poblahopřání hráči a přičtení a aktualizování bodů k dosavadnímu skóre hráče na základě obtížnosti.

4.2.4 Třída Resource

Třída `Resource` je zodpovědná za načítání a správu zdrojů (dat) pro hru. Konstruktor třídy přijímá parametr `difficulty` a na základě toho načítá odpovídající množství dat. Zdrojový soubor této třídy obsahuje přibližně 5000 slov. Díky takovému počtu je možné dosáhnout téměř neopakujících se dat k jednotlivým šifrám, což zaručuje jejich rozmanitost. Tato funkcionalita umožňuje využít bohatého slovníku a eliminovat opakování slov.

Výpis 4.7: Metoda pro načítání slov ze souboru

```
...
    public void loadWholeData() {
        cryptWholeRes = new ArrayList<>();
        Scanner sc = new Scanner(GameController.class.
            getResourceAsStream("data/wordVomit.txt"), "UTF-8"
            );
        while (sc.hasNext()) {
            String word = sc.next();
            cryptWholeRes.add(word);
        }
    }
    public void loadData(int difficulty) {
        Collections.shuffle(cryptWholeRes);
        cryptResource = new ArrayList<>();
        ListIterator<String> getThrowAll = cryptWholeRes.
            listIterator();
        while (getThrowAll.hasNext() && cryptResource.size()
            <= difficulty) {
            String word = getThrowAll.next();
            cryptResource.add(word);
        }
    }
...
```

Metody `loadWholeData` a `loadData` slouží k načítání celých dat a specifického množství dat pro danou obtížnost hry.

4.2.5 Abstraktní třída pro šifry

Třída `CiphersAbstract` je abstraktní třídou, která slouží jako základ pro implementaci různých šifer v kryptografické hře. Využitím abstraktní třídy se docílí jednotnost

a větší přehlednost kódu. Tato třída obsahuje hlavičky metod pro výpisy zadání, historii hádaných slov, zprávu po uhodnutí slova, nápovědu apod.

Výpis 4.8: Ukázka abstraktní třídy pro šifry

```
...
public void setKey(){
        int index = (int) Math.round(Math.random() * (res.
           cryptResource.size() - 1));
        key = res.cryptResource.get(index);
    }
    public abstract String task();
    public abstract String guess();
    public abstract String win();
    public abstract String hint();
    public abstract boolean solve(String answer);
    public abstract String encrypt();
    public abstract boolean insertAnswer(TextArea
       consoleOutTextArea, TextArea consoleInfTextArea,
       TextField consoleInTextField);
    public abstract void prepareOutArea(TextArea
       consoleOutTextArea);
...
```

Metoda `setKey` slouží k nastavení hodnoty atributu `key`. V této metodě se náhodně vybere index z rozsahu velikosti seznamu `res.cryptResource` (který je součástí instance třídy `Resource`) a přiřadí se odpovídající hodnota ze seznamu do atributu `key`. Tím se získá náhodně vybrané slovo pro šifrování. Tato třída definuje abstraktní metody `task`, `guess`, `win`, `hint`, `solve`, `encrypt`, `insertAnswer` a `prepareOutArea`, které slouží k implementaci konkrétních šifer a herní logiky v odvozených třídách.

4.2.6 Implementace šifry PasswordCrack

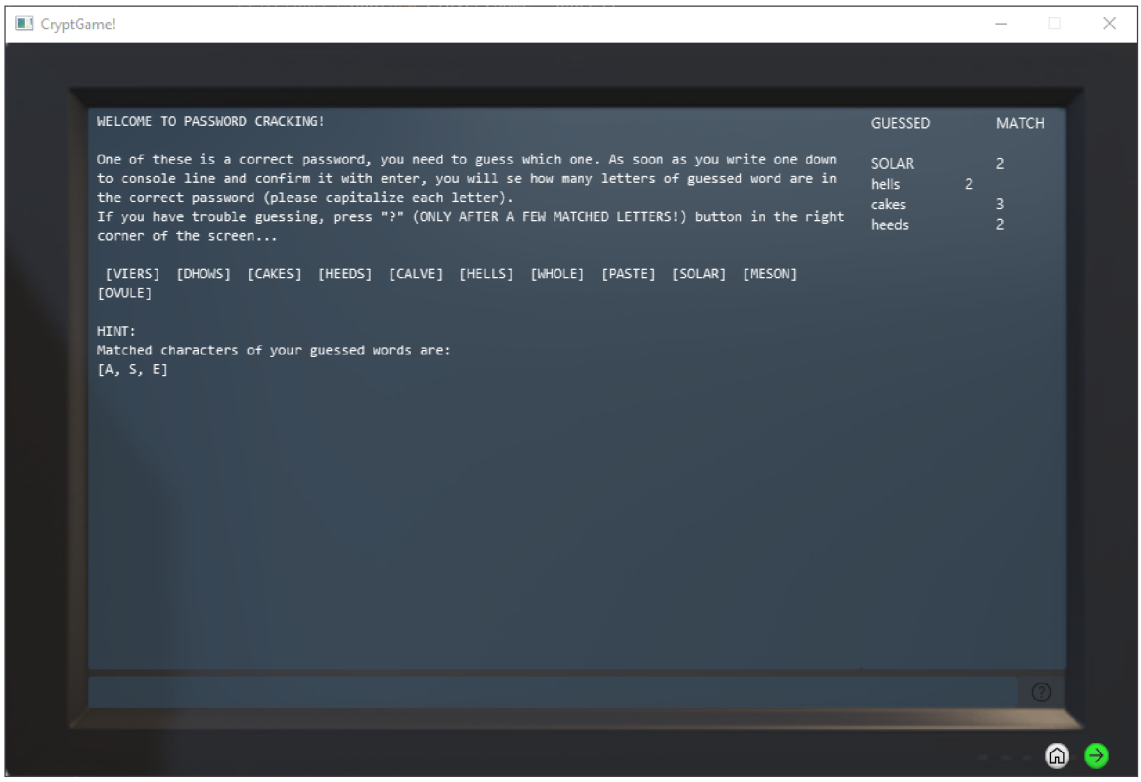
Šifra PasswordCrack není oficiální šifrou, proto je implementována mým vlastním způsobem. Třída `PasswdCrack` je odvozená třída od abstraktní třídy `CiphersAbstract` a představuje konkrétní implementaci šifry pro „crackování“ hesel v kryptografické hře. Principem této šifry je uhádnutí hledaného slova na základě shodných písmen se slovem, které zkusí zadat hráč. Aby se toto pro hráče ještě usnadnilo, je mu představeno několik slov, z nichž jedno je jistě to hledané.
Nejprve je zvolen počet nabízených slov dle obtížnosti (těžší obtížnosti znamená více nabízených slov) a jedno z těchto slov je zvoleno jako hledané `key`. Byla zde využita i vnořená třída `CharMatchPasswd` převážně z důvodu přehlednosti kódu. Vnořená

třída `CharMatchPasswd` slouží k nalezení shodujících se znaků mezi hádaným a hledaným slovem. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.4.

Výpis 4.9: Ukázka funkce pro kontrolu shodných znaků slov

```
...
public CharMatchPasswd findCharMatch(TextArea
   consoleInfTextArea, TextField consoleInTextField) {
        charTemp.clear();
        if (key.length() == consoleInTextField.getText().
           length()) {
            for (int i = 0; i < key.length(); i++) {
                char charPasswd = key.charAt(i);
                for (int j = 0; j < consoleInTextField.
                   getText().length(); j++) {
                    char charIn = consoleInTextField.getText
                       ().charAt(j);
                    charIn = Character.toUpperCase(charIn);
                    if (charPasswd == charIn) {
                        charTemp.add(charPasswd); //přidání
                           znaku do Setu shodných znaků
                        break;
                    }
...
```

Funkce `findCharMatch` slouží k nalezení shodujících se znaků mezi hádaným slovem (získaným z uživatelského vstupu) a tím hledaným. Nejprve se vyčistí množina `charTemp`. Poté se porovnávají znaky mezi oběma řetězci, přičemž se postupně projdou znaky v obou řetězcích a pokud je nalezena shoda, přidá se tento znak do množiny `charTemp`. Nakonec se shodující znaky přidají do množiny `characters` a vytvoří se instance třídy `CharMatchPasswd` s informacemi o počtu shodujících se znaků a množinou shodujících se znaků. Tyto shodné znaky se nakonec vypíší do GUI.



Obr. 4.4: Snímek z šifry PasswordCrack

4.2.7 Implementace Morseovy šifry

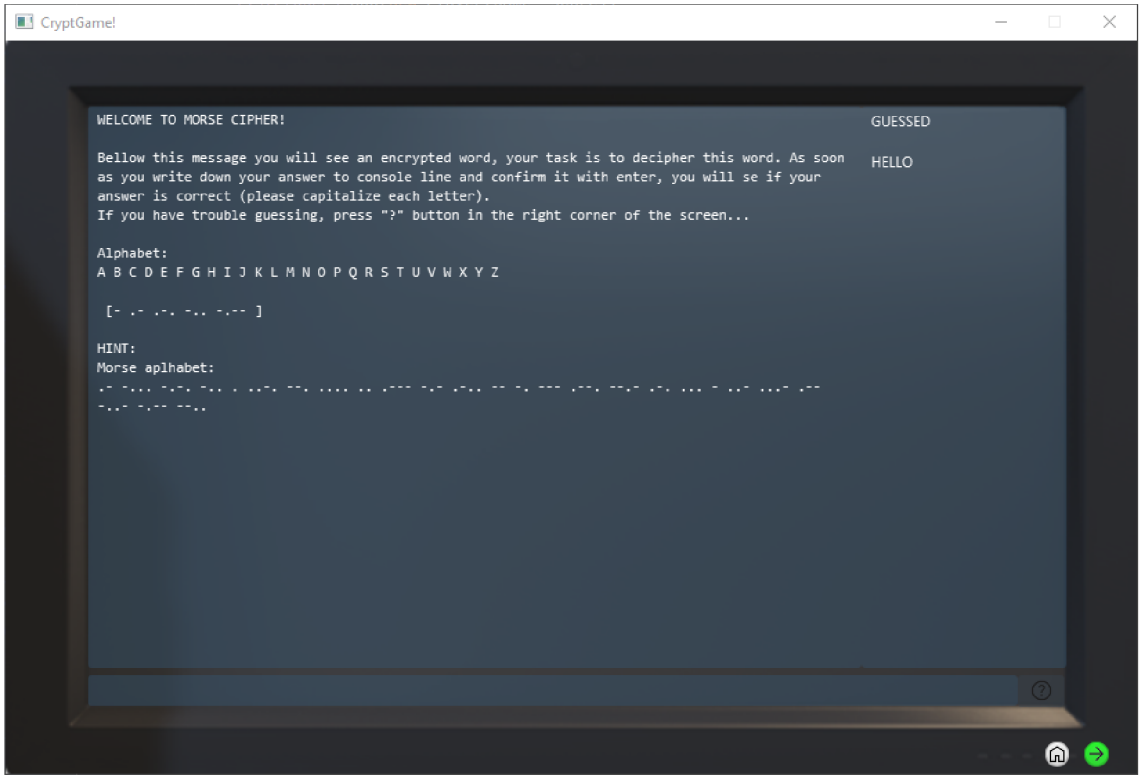
Třída `MorseCipher` je potomkem abstraktní třídy `CiphersAbstract` a implementuje šifru Morseova kódu (viz 2.2.1). Obsahuje metody pro generování abecedy v Morseově kódu, převod znaků do Morseova kódu a zpět, šifrování a dešifrování zpráv v Morseově kódu. Třída obsahuje také privátní metody pro generování Morseovy abecedy, zobrazení abecedy v textové podobě, převod Morseovy abecedy na řetězec a metodu pro šifrování vstupního slova do Morseova kódu. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.5.

Výpis 4.10: Ukázka funkce pro zašifrování slova

```
...
    @Override
    public String encrypt() {
        String s = "";
        for (int i = 0; i < key.length(); i++) {
            char ch = key.charAt(i);
            String morse = alphabet.get(ch);
            s += morse + " ";
```

```
9           }
10          return s;
11      }
12  ...
```

Metoda `encrypt` slouží k převedení vstupního slova (řetězce) do Morseova kódu. Pro každý znak ve slově se získá příslušný Morseův kód z abecedy (`alphabet`) a tyto kódy se postupně spojí dle indexů znaků do výsledného řetězce. Tím se dosáhne šifrování klíče do Morseova kódu. Výsledný Morseův kód je pak vrácen jako výstup metody.



Obr. 4.5: Snímek z Morseovy šifry Morse

4.2.8 Implementace Caesarovy šifry

Třída `CaesarCipher` implementuje Caesarovu šifru (viz 2.2.2) jako jednu z variant šifer v rámci aplikace. Tato třída rozšiřuje abstraktní třídu `CiphersAbstract` a implementuje její abstraktní metody. Pracuje se zde s klasickou abecedou (`alphabet`) vytvořenou jako mapa znaků. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.6.

Výpis 4.11: Ukázka funkce pro generování náhodného celého čísla

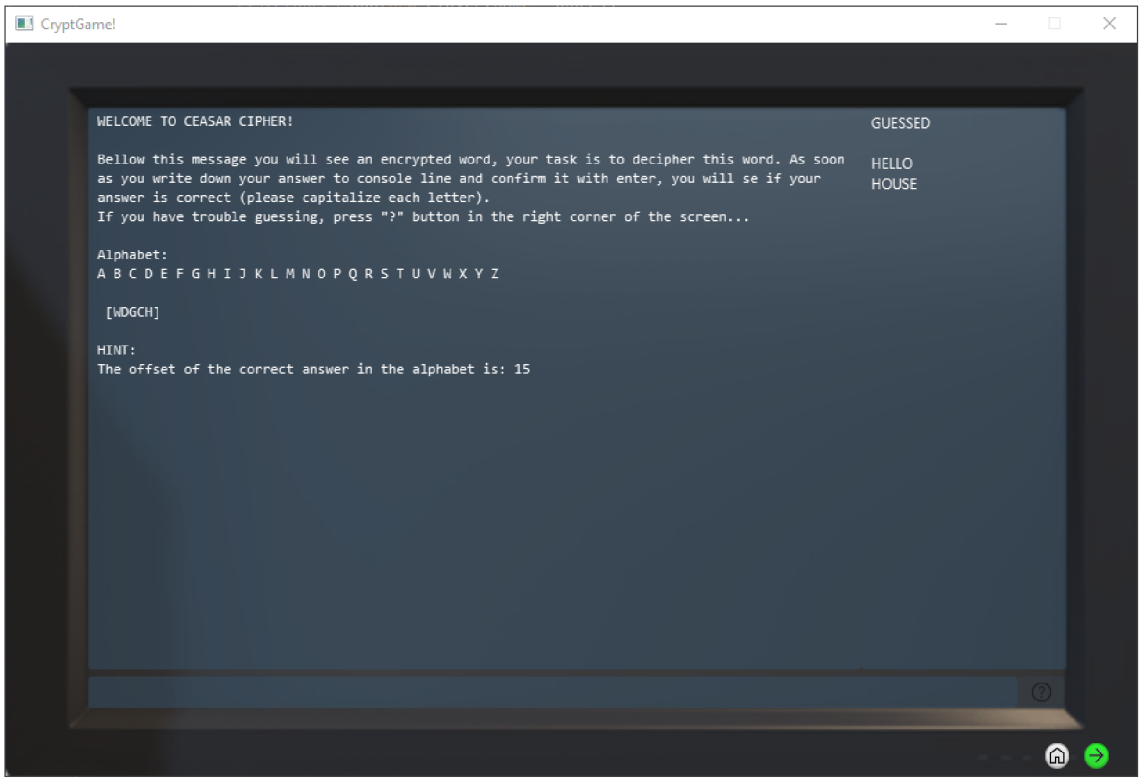
```
...
    private int generateOffset() {
        Random r = new Random();
        int offset = r.nextInt(25 - 1) + 1;
        return offset;
    }
...
```

Metoda `generateOffset` generuje náhodné číslo pro posunutí abecedy v Caesarově šifře (`offset`). Používá se pro určení, o kolik míst se budou jednotlivé znaky šifrovat. Metoda vytváří instanci třídy `Random` a generuje náhodné číslo v rozsahu od 1 do 25 (vyjma). Pokud by byla hodnota posunu 25, nedojde k žádnému posunu písmen, a písmena zůstanou na svých původních pozicích, což by znamenalo, že šifrované slovo by nebylo zašifrováno.

Výpis 4.12: Ukázka funkce pro posunutí znaků v abecedě

```
...
    @Override
    public String encrypt() {
        String encrypt = "";
        for (int i = 0; i < key.length(); i++) {
            char object = key.charAt(i);
            int ascii = object + offset;   //posun každého pí
               smene dle hodnoty posunu
            if (ascii > 90) {   //oprava přetečení abecedy
                int dif = ascii - 90 - 1;
                ascii = 65 + dif;
            }
            encrypt += (char) ascii;
        }
        return encrypt;
    }
...
```

Metoda `encrypt` prochází postupně jednotlivé znaky klíče a provádí jejich posunutí v abecedě o hodnotu `offsetu`. k tomu se nejprve získá ASCII hodnota daného znaku, k níž se přičte hodnota `offsetu`. Pokud tato hodnota přesahuje ASCII hodnotu „Z“ (90), dochází ke přetečení abecedy, a znovu se začíná od „A“ (65). Výsledné posunuté znaky jsou postupně přidávány do šifrovaného řetězce. Nakonec je šifrované slovo vráceno jako návratová hodnota metody.



Obr. 4.6: Snímek z Caesarovy šifry

4.2.9 Implementace šifry Atbash

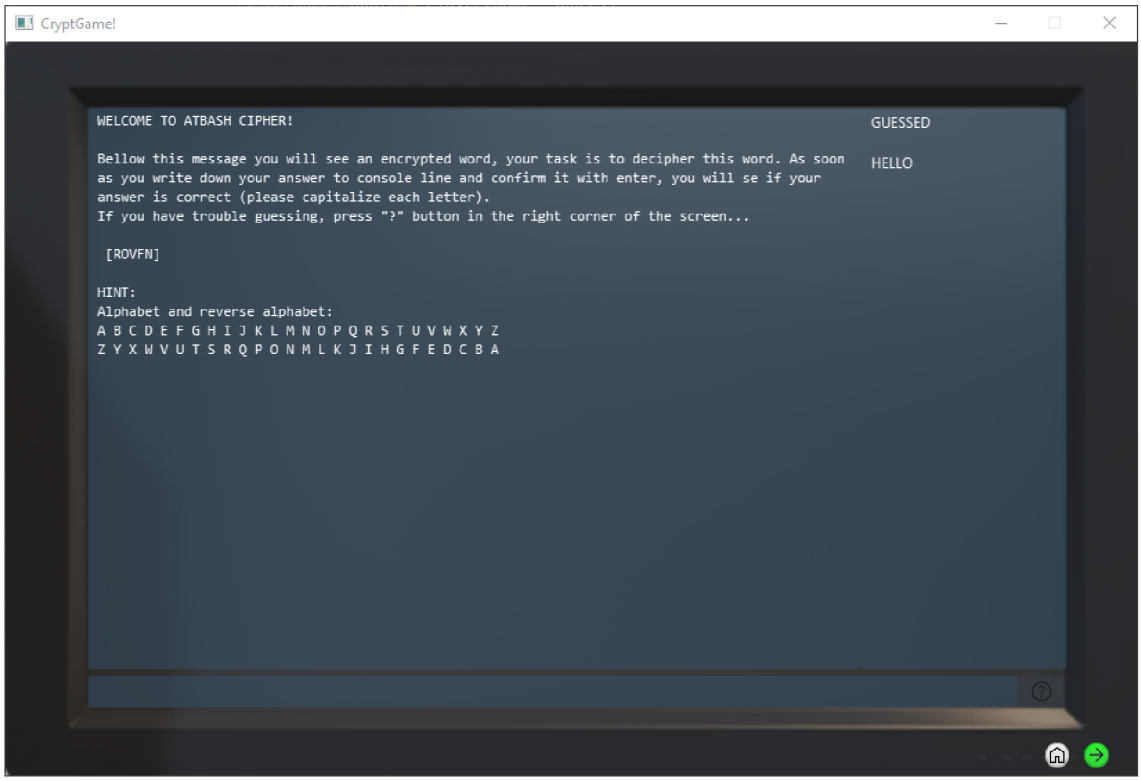
Třída `AtbashCipher` je potomkem třídy `CiphersAbstract` a implementuje šifru Atbash (viz 2.2.2). Tato třída obsahuje metody pro generování abecedy, šifrování, zobrazování abecedy a reverzní abecedy. Ve třídě se nachází `ArrayList` s názvem `alphabet`, který obsahuje písmena klasické abecedy. `ArrayList` je využit z důvodu snadné manipulace s pořadím a tříděním obsahu. Nebylo třeba zde proto vytvářet nové obrácené abecedy, ale stačilo zavolat nad objektem `Collection` funkci `reverse` s parametrem `alphabet`. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.7.

Výpis 4.13: Ukázka funkce užívající obrácené abecedy k zašifrování hledaného slova

```
1 ...
2     @Override
3     public String encrypt() {
4         String encrypt = "";
5         ArrayList<Character> orig = new ArrayList<>(alphabet)
              ;
6         Collections.reverse(alphabet);
```

```
7           for (int i = 0; i < key.length(); i++) {
8               Character character = key.charAt(i);
9               int index = orig.indexOf(character);
10              encrypt += alphabet.get(index);
11          }
12          return encrypt;
13      }
14  ...
```

Metoda `encrypt` nejprve vytváří kopii seznamu abecedy (`alphabet`) a následně tento seznam obrátí. Poté projde každý znak ve vstupním slovu a hledá jeho index v původním seznamu abecedy. Na základě indexu přistupuje ke stejnému indexu v seznamu obrácené abecedy a přiřazuje odpovídající znak. Nakonec vytvoří zašifrovaný text spojením těchto znaků do jednoho řetězce, který vrátí jako výstup metody.



Obr. 4.7: Snímek z šifry Atbash

4.2.10 Implementace šifry Polybius

Třída `PolybiusCipher` implementuje Polybiusovu šifru (viz 2.2.5) jako podtřídu třídy `CiphersAbstract`. Při inicializaci třídy se vytvoří dvourozměrné pole `alphabet`

obsahující písmena abecedy organizovaná podle Polybiusovy šifry. Poté se vygeneruje zašifrovaný text pomocí metody `encrypt`, která projde každý znak vstupního textu a nahrazuje ho souřadnicemi v rámci dvourozměrného pole `alphabet`. Třída obsahuje také metody pro zobrazení abecedy a úkolového textu, kontrolu řešení, zobrazení nápovědy a ovládání vstupu a výstupu z konzole. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.8.

Výpis 4.14: Ukázka funkce generující mřížku abecedy

```
...
   private char[][] setAlphabet() {
        char[][] alphabet = new char[5][5];
        int ascii = 65;
        for (int i = 0; i < alphabet.length; i++) {
            for (int j = 0; j < alphabet[i].length; j++) {
                char ch = (char) ascii++;
                if (ascii != 74) {  // pro přeskočení znaku J
                    char c = alphabet[i][j] = ch;
                } else {
                    char c = alphabet[i][j] = 73;
                    ascii++;
...
```

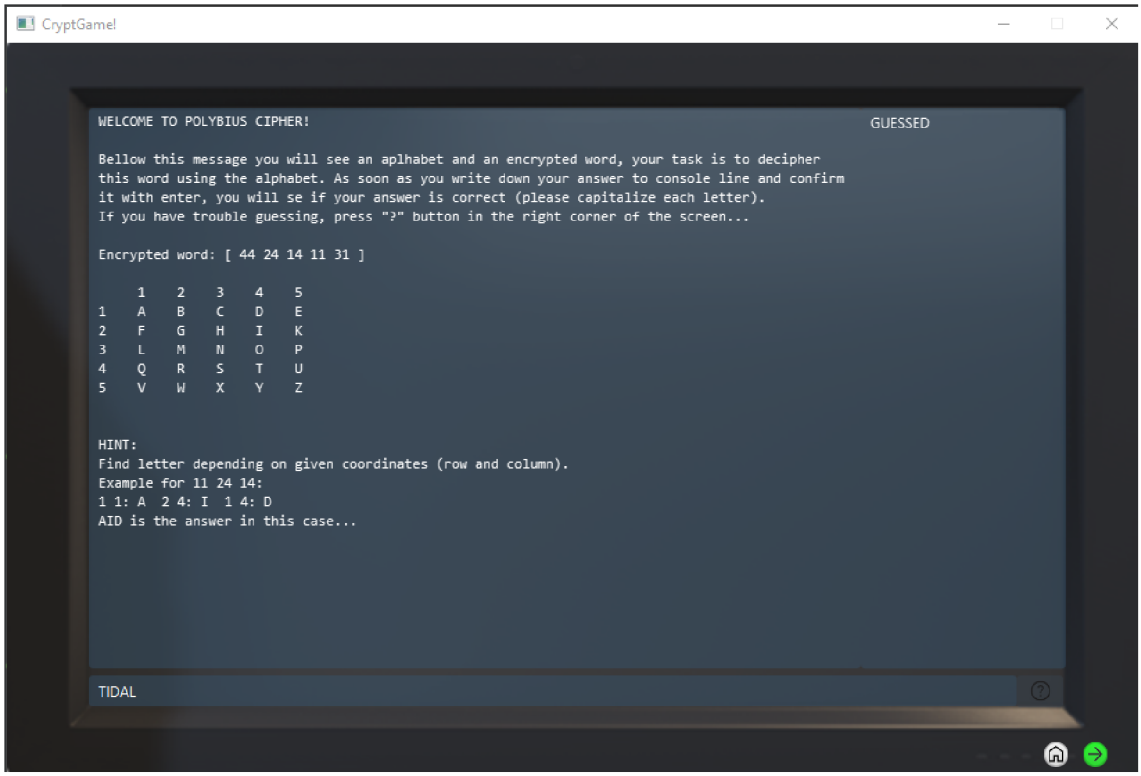
Metoda `setAlphabet` slouží k inicializaci dvourozměrného pole `alphabet`, které reprezentuje abecedu používanou v Polybiusově šifře. Metoda postupně prochází každý řádek a sloupec pole `alphabet` a přiřazuje do každé buňky odpovídající znak abecedy. Inicializace probíhá tak, že se postupně zvyšuje hodnota proměnné `ascii` od 65 (která odpovídá ASCII hodnotě písmene „A“). Pokud je hodnota `ascii` rovna 74 (což odpovídá písmenu „J“ v ASCII), tak se místo toho přiřadí hodnota 73, aby se přeskočilo písmeno „J“ v Polybiusově šifře. Nakonec je vráceno inicializované dvourozměrné pole `alphabet`.

Výpis 4.15: Ukázka funkce pro zašifrování dle mřížky

```
...
    @Override
    public String encrypt() {
        String encrypedKey = "";
        for (int i = 0; i < key.length(); i++) {
            char ch = key.charAt(i);
            for (int j = 0; j < alphabet.length; j++) {
                char[] cs = alphabet[j];
                for (int k = 0; k < cs.length; k++) {
                    char c = cs[k];
```

```
11                          if (ch == c) {
12                              encrypedKey += (j + 1) + "" + (k + 1)
                                    + " ";
13                              break;
14  ...
```

Metoda `encrypt` slouží k zašifrování vstupního textu pomocí Polybiusovy šifry. Prochází postupně každý znak ve vstupním textu. Poté pro každý znak hledá jeho odpovídající souřadnice v dvourozměrném poli `alphabet`, které obsahuje písmena abecedy organizovaná podle Polybiusovy šifry. Souřadnice znaku jsou reprezentovány číslem řádku a číslem sloupce v poli `alphabet`, ale jelikož souřadnice pole začínají na 0, tak je nutné ke každé souřadnici přičíst hodnotu 1. Po nalezení správných souřadnic se tyto souřadnice připojí k zašifrovanému textu `encryptedKey` ve formátu „řádek sloupec“. Nakonec je vrácen zašifrovaný text.



Obr. 4.8: Snímek z šifry Polybius

4.2.11 Implementace Vigenèrovy šifry

Třída `VigenereCipher` představuje implementaci Vigenèrovy šifry (viz 2.2.3), která je odvozená od abstraktní třídy `CiphersAbstract`. Obsahuje dvourozměrné pole

`alphabet`, které reprezentuje abecedu pro Vigenèrovu šifru, a proměnnou `mask`, která je maskovacím slovem (klíčem). Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.9.

Výpis 4.16: Ukázka funkce generující mřížku abecedy

```
...
    private char[][] setAlphabet() {
        char[][] alphabet = new char[26][26];
            int ascii = 65;
            int start = 65;
            for (int i = 0; i < alphabet.length; i++) {
                ascii = start++;      //začítek abecedy (65 =
                   A, poté 66 = B,...)
                for (int j = 0; j < alphabet[i].length; j++)
                   {
                    if (ascii > 90) {
                        int dif = ascii - 90;
                        ascii = 65 + dif - 1;
                    }
                    char c = alphabet[i][j] = (char) ascii++;
...
```

Metoda `setAlphabet` slouží k vytvoření abecedy pro Vigenèrovu šifru. Tato metoda vytváří dvourozměrné pole `alphabet` o rozměrech 26x26, které reprezentuje abecedu. Postupně projde každý řádek a sloupec v poli `alphabet` a přiřadí jim odpovídající znaky abecedy. Začíná s ASCII hodnotou 65, která představuje znak „A“. Každý další řádek se posouvá o jednu ASCII hodnotu výše. Pokud přiřazená ASCII hodnota přesáhne 90 (která představuje znak „Z“), provede se cyklické procházení abecedou. Vypočte se rozdíl mezi aktuální hodnotou a 90 a k tomuto rozdílu se přičte 64, čímž se získá nová ASCII hodnota. Na závěr je vytvořená abeceda vrácena jako výsledek metody.

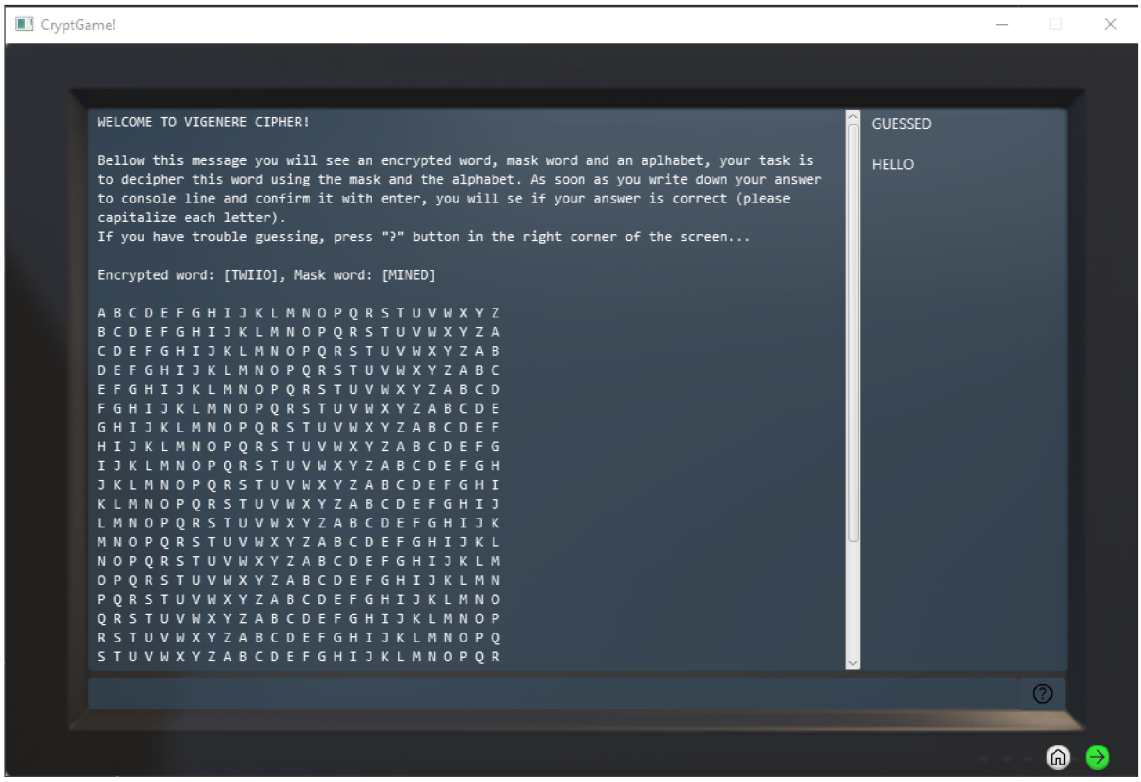
Výpis 4.17: Ukázka funkce pro šifrování Vigenèrovy šifry

```
...
    @Override
    public String encrypt() {
        String encrypedKey = "";

        for (int i = 0; i < mask.length(); i++) {
            char r = mask.charAt(i);
            int row = r - 65;
            int column = 0;
```

```
10              for (int j = 0; j < key.length(); j++) {
11                  char ch = key.charAt(i);
12                  column = ch - 65;
13              }
14              encrypedKey += alphabet[row][column];
15          }
16          return encrypedKey;
17      }
18  ...
```

Metoda `encrypt` slouží k zašifrování vstupního slova (klíče) pomocí Vigenèrovy šifry. Šifrování se provádí postupně pro každý znak v klíči a maskovacím řetězci. Pro každý znak v klíči se získá jeho odpovídající znak v prvním řádku tabulky abecedy (`alphabet`). Následně se pro každý znak v maskovacím řetězci získá jeho odpovídající znak v prvním sloupci tabulky abecedy. Na základě řádku a sloupce v tabulce se vybere průsečík, který tvoří zašifrovaný znak. Tento zašifrovaný znak se postupně přidává do výsledného zašifrovaného klíče. Po projití všech znaků v klíči je výsledkem metody zašifrovaný klíč, který je vrácen jako výstup.



Obr. 4.9: Snímek z Vigenèrovy šifry

4.2.12 Implementace šifry Bifid

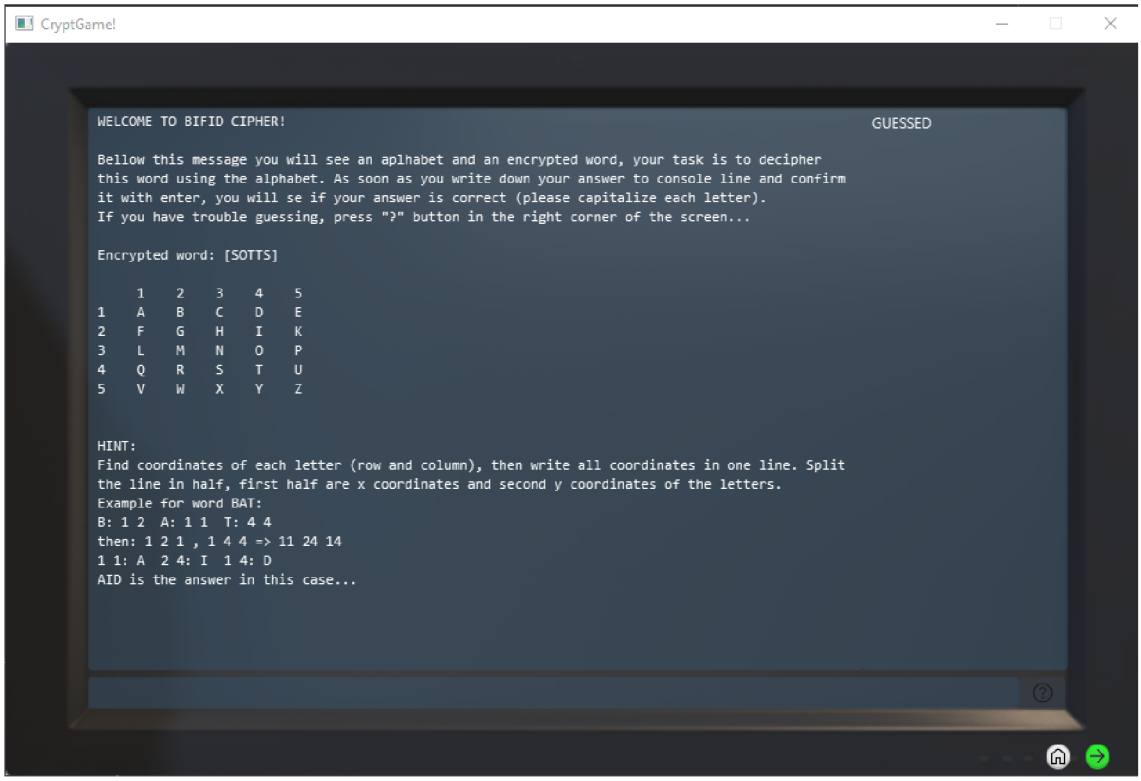
Třída `BifidCipher` je implementací Bifid šifry (viz 2.2.3), která dědí od abstraktní třídy `CiphersAbstract`. Obsahuje atribut dvojrozměrné pole znaků `alphabet`, který představuje abecedu používanou pro šifrování. Vytváření dvourozměrného pole `alphabet` je realizováno stejně jako u implementace šifry Polybius (viz 4.2.10. Výsledné zobrazení této šifry přímo ve hře je vidět na obrázku 4.10.

Výpis 4.18: Ukázka funkce rozdělující pole na sudé a liché a následné šifrování

```
...
     @Override
    public String encrypt() {
    ...
        int[] field1D = convert2DTo1D(field2D);
        // rozdělení pole na liché a~sudé
        int[] odd = new int[field1D.length / 2];
        int o = 0;
        for (int i = 0; i < field1D.length; i++) {
            int c = field1D[i];
            if (i % 2 == 1) {
                odd[o++] = c;
            }
        }
        int[] even = new int[field1D.length / 2];
        int e = 0;
        for (int i = 0; i < field1D.length; i++) {
            int c = field1D[i];
            if (i % 2 == 0) {
                even[e++] = c;
            }
        }
        //zpětné šifrování dle mřížky
        for (int i = 0; i < even.length; i++) {
            int ev = even[i];
            int od = odd[i];
            encrypedKey += alphabet[ev][od];
        }
        return encrypedKey;
    }
...
```

Metoda `encrypt` je implementací šifrování v rámci Bifid šifry. Nejprve se vytváří

dvourozměrné pole `field2D`, které slouží k uchování souřadnic jednotlivých znaků v abecedě. Poté se toto pole převádí na jednorozměrné pole `field1D`. Následně se toto pole rozděluje na pole `odd` a `even` na základě lichých a sudých indexů. Šifrování probíhá postupným vybíráním písmen na základě lichých a sudých souřadnic z pole `odd` a `even` s tím, že sudé určují souřadnici x a liché souřadnici y. Vybraná písmena se postupně přidávají k výslednému zašifrovanému klíči. Nakonec je vrácen výsledný zašifrovaný klíč jako textový řetězec.



Obr. 4.10: Snímek z šifry Bifid

4.2.13 Implementace herního skóre

Třída `GameScore` představuje implementaci skóre a správy hráčů ve hře. Obsahuje statickou vnitřní třídu `Player`, která reprezentuje hráče s jejich jménem a skóre. Hráči jsou ukládáni v kolekci `listPlayers`, která je implementována jako seznam. Třída umožňuje přidávat hráče, nastavovat jejich skóre, řadit hráče podle skóre, zobrazovat seznam hráčů, ořezávat seznam na deset nejlepších a provádět načítání a ukládání skóre ze souboru. Výsledné zobrazení seznamu se skóre je vidět na obrázku 4.11.

Výpis 4.19: Ukázka funkce pro přidání nového hráče

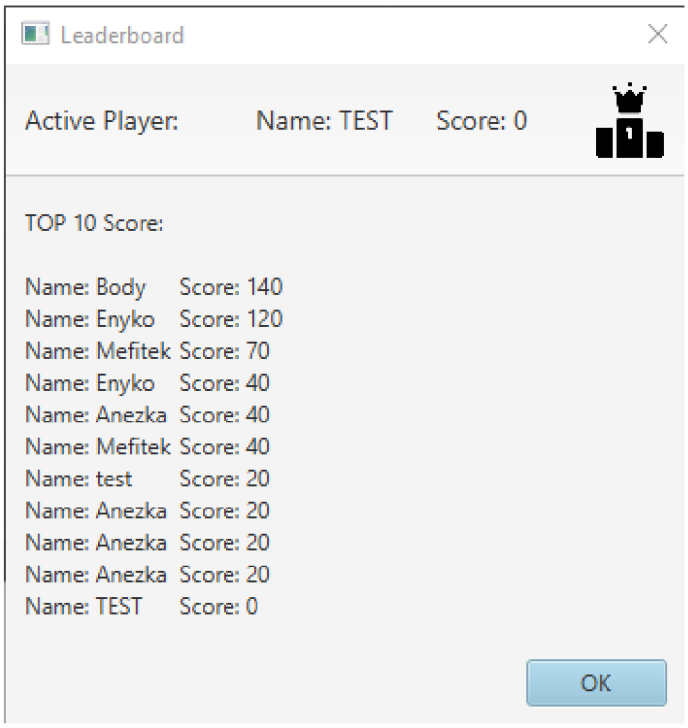
```
...
    static void addPlayer(String name, int score) {
        active = new Player(name, score);
        listPlayers.add(active);
    }
...
```

Funkce `addPlayer` slouží k přidání nového hráče do seznamu hráčů `listPlayers`. Vytváří novou instanci třídy `Player` s daným jménem a skóre a přidá ji do seznamu hráčů. Tato funkce umožňuje dynamicky přidávat nové hráče do seznamu a ukládat jejich jméno a skóre pro pozdější použití a správu.

Výpis 4.20: Ukázka funkce pro výběr aktuálního hráče

```
...
    static int selectPlayer(String name) {
        int index = -1;
        for (int i = 0; i < listPlayers.size(); i++) {
            Player get = listPlayers.get(i);
            if (get.name.equalsIgnoreCase(name)) {
                index = i;
            }
        }
        return index;
    }
...
```

Funkce `selectPlayer` slouží k vyhledání hráče v seznamu hráčů `listPlayers` na základě zadaného jména. Prochází seznam hráčů a porovnává zadané jméno s jmény hráčů v seznamu. Pokud najde shodu, vrátí index tohoto hráče v seznamu. Pokud nebyl hráč nalezen, vrací hodnotu -1. Tato funkce umožňuje vyhledávat hráče v seznamu na základě jejich jména a poskytuje možnost identifikovat konkrétního hráče pro další operace, například aktualizaci jeho skóre.



Obr. 4.11: Snímek z tabulky skóre

Závěr

První část práce se věnuje teorii kryptologie. Kryptologie vznikla spojením kryptografie, kryptoanalýzy a steganografie a jelikož spolu tyto vědy úzce souvisejí, nelze se věnovat jen jedné z nich.
V druhé části se už zaměřujeme spíše na moderní a klasickou kryptografii. Jsou zde uvedeny typické příklady z každé skupiny šifer.
Třetí část se zabývá technologiemi využitými při tvorbě praktické části projektu. Jako programovací jazyk byla vybrána Java a pro tvorbu uživatelského rozhraní JavaFX.
Poslední část projektu je už čistě zaměřena na praktickou stránku práce. Je zde uveden postup, podle kterého byla vypracována bakalářská práce. Nejprve byl vytvořen návrh uživatelského rozhraní v programu Scene Builder a výsledný fxml soubor byl implementován do kódu. První návrh GUI má účel menu, neboli okna, ze kterého bude možné ovlivnit možnosti hry. Druhý návrh je poté již zaměřen na samotnou hru. Design byl zamýšlen zpodobňovat počítačovou konzoli, aby hráči navodil správnou atmosféru. Sem se nakonec vykreslují data a hráč má za úkol vyřešit nabízené šifry. Složitost a typy šifer jsou ovlivněny obtížností, kterou hráč zvolí na začátku v menu. Jako nápověda ve hře slouží tlačítko, po jehož stisknutí se objeví individuální nápověda. Hráči to poté pomůže vydedukovat správnou odpověď. Po dokončení šifry je hráči pogratulováno a může buďto pokračovat na další šifru, nebo se vrátit do menu a zvolit si jinou obtížnost či se podívat na své skóre.
Práce splnila cíl vytvořit interaktivní vzdělávací hru s přívětivým uživatelským rozhraním. Může být dále využita například k interaktivní prezentaci kryptologie a oboru Informační bezpečnost na dni otevřených dveří.
Rozšíření práce by bylo možné přidáním více šifer, kterými se může hráč zabývat. Další možností vylepšení je umožnění změny velikosti oken a distribuce hry na web.

Literatura

[1] VONDRUŠKA, P. *Kryptologie, šifrování a tajná písma*. 1. vyd. Praha: Albatros, 2006 [cit. 25. 11. 2022]. ISBN: 8000018888.

[2] HAVRLANT, L. *Kryptografie & šifrování*. Matematika polopatě, článek, [online] 2022. [cit. 5. 12. 2022]. Dostupné z URL: `https://www.matweb.cz/kryptografie/`.

[3] VACEK, J. *KRYPTOGRAFIE*. [online] 2001. [cit. 5. 12. 2022]. Dostupné z URL: `https://www.kip.zcu.cz/kursy/svt/svt_www/3_soubory/3_5.htm`.

[4] *dCode's Tools List on dCode.fr* [online]. poslední aktualizace 2022 [cit. 25. 11. 2022]. Dostupné z URL: <`https://www.dcode.fr/tools-list`>.

[5] RIVEST, R., SHAMIR, A., ADLEMAN L. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM, Vol. 21 (2), 1977 [cit. 25. 11. 2022].

[6] CORMEN, T., LEISERSON, C., RIVEST, R., STEIN, C. *Introduction to Algorithms*. Second Edition, MIT Press and McGraw-Hill, 2001. [cit. 25. 11. 2022]. ISBN: 0-262-03293-7. Section 31.7: The RSA public-key cryptosystem

[7] *Kódování a šifrování*. Stránky k výuce Informatiky, [online] 2022. [cit. 6. 12. 2022]. Dostupné z URL: `http://www.ivt.mzf.cz/informatika-v-kostce/zakladni-pojmy/kodovani-a-sifrovani/`.

[8] *Braillovo písmo*. Svetabeced.cz Web o abecedách, písmu a znacích, [online] 2015. [cit. 6. 12. 2022]. Dostupné z URL: `http://www.svetabeced.cz/ostatni/braillovo-pismo/`.

[9] MIKLÁŠ, M. *Kódování textu a znakové sady 1*. www.gjszlin.cz, [online] 2022. [cit. 6. 12. 2022]. Dostupné z URL: `https://www.gjszlin.cz/ivt/esf/ostatni-sin/kodovani-textu-1.php/`.

[10] *Morseovka*. Svetabeced.cz Web o abecedách, písmu a znacích, [online] 2015. [cit. 6. 12. 2022]. Dostupné z URL: `http://www.svetabeced.cz/ostatni/morseovka/`.

[11] PODSTATA, J. *Kryptoanalýza klasických šifer* [online] 2016. Bakalářská práce. Univerzita Palackého v Olomouci, Přírodovědecká fakulta. Vedoucí práce RNDr. Eduard Bartl, Ph.D. [cit. 7. 12. 2022]. Dostupné z URL: `https://theses.cz/id/spahu1/`.

[12] HAVRLANT, L. *Vigenèrova šifra*. Matematika polopatě, článek, [online] 2022. [cit. 8. 12. 2022]. Dostupné z URL: https://www.matweb.cz/vigenerova-sifra/.

[13] *Vigenere Cipher*. dcode, [online] 2023. [cit. 22. 05. 2023]. Dostupné z URL: <https://www.dcode.fr/vigenere-cipher>.

[14] VONDRUŠKA, P. *Šifra Delastelle - BIFID*. root.cz, [online] 2006. [cit. 21. 05. 2023]. Dostupné z URL: https://www.root.cz/clanky/sifra-delastelle-bifid/.

[15] *Bifid Cipher*. dcode, [online] 2023. [cit. 21. 05. 2023]. Dostupné z URL: <https://www.dcode.fr/bifid-cipher>.

[16] Shaman. *Šifra Playfair*. Shaman.cz tak trochu jinou cestou, [online] 2021. [cit. 7. 12. 2022]. Dostupné z URL: http://www.shaman.cz/sifrovani/sifra-playfair.htm.

[17] HOUSER, P. *Co je to Playfairova šifra?*. ScienceWORLD, [online] 2006. [cit. 8. 12. 2022]. Dostupné z URL: https://www.scienceworld.cz/neziva-priroda/co-je-to-playfairova-sifra-1245/.

[18] *Polybius Cipher*. dcode, [online] 2023. [cit. 21. 05. 2023]. Dostupné z URL: <https://www.dcode.fr/polybius-cipher>.

[19] Rodriguez-Clark, D. *Cryptography Worksheet — Polybius Square*. PDF, [online]. [cit. 23. 05. 2023]. Dostupné z URL: http://crypto.interactive-maths.com/uploads/1/1/3/4/11345755/polybius_square.pdf.

[20] Rodriguez-Clark, D. *Pigpen Cipher*. Crypto Corner, [online] 2022. [cit. 21. 05. 2023]. Dostupné z URL: https://crypto.interactive-maths.com/pigpen-cipher.html.

[21] SEBESTA, R. W. *Concepts of programming languagess*. Addison-Wesley, 2010. [cit. 8. 12. 2022]. ISBN: 01-360-7347-6.

[22] KUBŮ, P. *Úvod do programovacího jazyka Java*. Petr Kubů Homepage, [online] 2022. [cit. 8. 12. 2022]. Dostupné z URL: http://kubu.wz.cz/help/java/java.html.

[23] HARTMAN, J. *What is Java? Definition, Meaning & Features of Java Platforms*. Guru99, [online] 2022. [cit. 9. 12. 2022]. Dostupné z URL: https://www.guru99.com/java-platform.html.

[24] *JavaFX: Getting Started with JavaFX*. Oracle, [online] 2014. [cit. 9. 12. 2022]. Dostupné z URL: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784.

[25] *JavaFX Scene Builder Informations*. Oracle, [online] 2022. [cit. 9. 12. 2022]. Dostupné z URL: https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html.

[26] pablinux. *NetBeans: co to je a jak jej nainstalovat na Ubuntu a deriváty*. root.cz, [online] 2022. [cit. 8. 12. 2022]. Dostupné z URL: https://ubunlog.com/cs/netbeans/.

Seznam symbolů a zkratek

3D Three Dimensional

AES Advanced Encryption Standard

ASCII American Standard Code for Information Interchange

DES Data Encryption Standard

GUI Graphic User Interface

ID Identify Document

IDE Integrated Development Environment

Java EE Java Platform, Enterprise Edition

Java ME Java Platform, Mobile Edition

Java SE Java Platform, Standard Edition

JDK Java Development Kit

JRE Java Runtime Environment

RC4 Rivest Cipher 4

RSA iniciály autorů Rivest, Shamir, Adleman

A Obsah elektronické přílohy

- `README.md` ..... soubor README
- `src/` ..... kořenový adresář přiloženého archivu
  - `com/`
    - `project/`
      - `cryptgame/`
        - `AtbashCipher.java` ..... třída pro šifru Atbash
        - `BifidCipher.java` ..... třída pro šifru Bifid
        - `CaesarCipher.java` ..... třída pro Caesarovu šifru
        - `CiphersAbstract.java` ..... abstraktní třída pro šifry
        - `CryptGameMain.java` ..... spouštěcí soubor
        - `GameController.java` ..... controller pro GUI hry
        - `GameScore.java` ..... třída pro skóre
        - `MenuController.java` ..... controller pro GUI menu
        - `MorseCipher.java` ..... třída pro Morseovu šifru
        - `PasswdCrack.java` ..... třída pro PasswordCrack šifru
        - `PolybiusCipher.java` ..... třída pro šifru Polybius
        - `Resource.java` ..... třída pro zdroje
        - `VigenereCipher.java` ..... třída pro Vigenerovu šifru
  - `res/`
    - `css/`
      - `style.css` ..... css soubor pro vzhled oken
    - `data/`
      - `dataScore.txt` ..... textový soubor obsahující seznam hráčů a jejich skóre
      - `wordWomit.txt` ..... textový soubor obsahující seznam slov pro hru
    - `pictures/`
      - `computer.png` ..... obrázek využitý jako pozadí pro menu
      - `gameConsole.png` ..... obrázek využitý jako pozadí pro hru
      - `hint.png` ..... obrázek využitý jako ikona pro radu
      - `leaderboard.png` ..... obrázek využitý jako ikona pro tabulku skóre
      - `menu.png` ..... obrázek využitý jako ikona pro tlačítko do menu
      - `next.png` ..... obrázek využitý jako ikona pro tlačítko pro další šifru
    - `cryptgame.fxml` ..... soubor obsahující grafický návrh hry
    - `cryptmenu.fxml` ..... soubor obsahující grafický návrh menu

Aktuální řešení je dostupné na GitHubu: `https://github.com/xfisar02/CryptGame`