STU FIIT

Fakulta informatiky a informačných technológii

Ilkovičova 2, 842 16 Bratislava 4

Simulácia zneškodnenia bomby

(Projekt)

Mária Matušisková
ZOOP
Cvičiaca: Bc. Alexandra Skyvová (Ut 10:00)

ZS 2022/2023

Rámcové zadanie

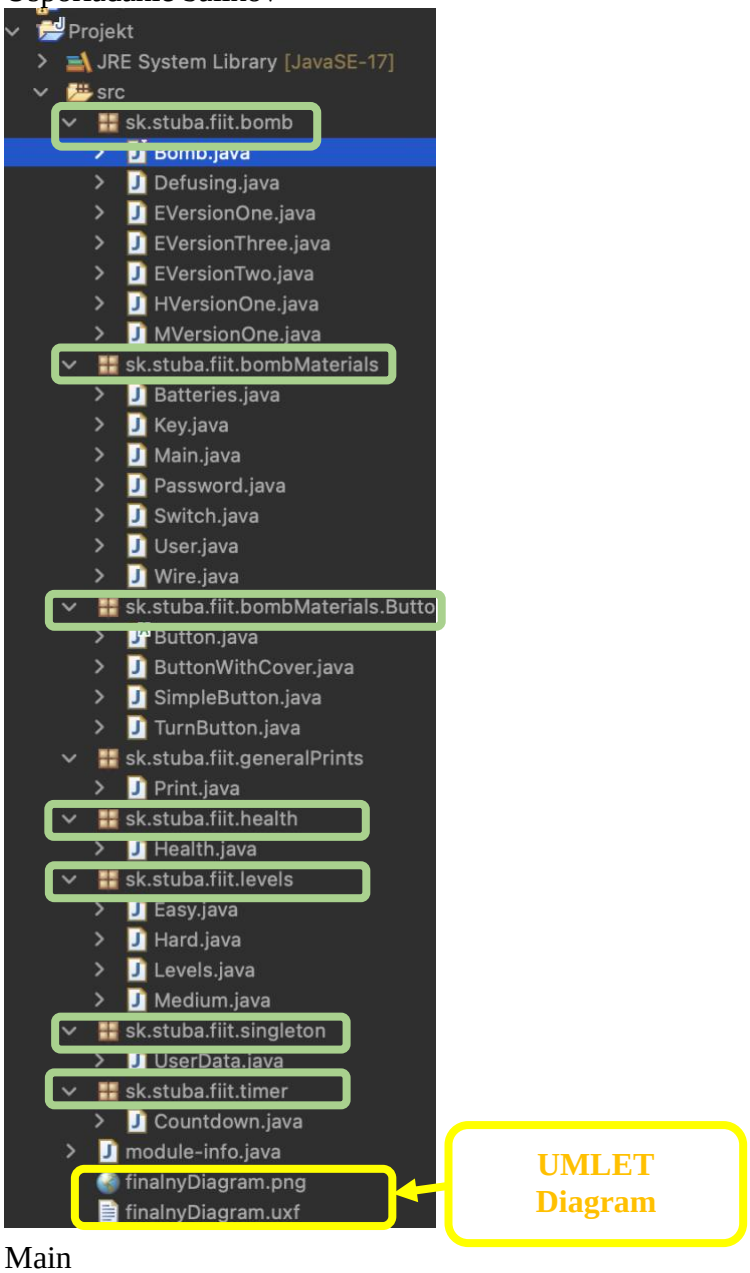
Ako rámcovú tému sme si zvolili Simuláciu zneškodnenia bomby pre vzdelávacie a herné účely. Hra bude pozostávať z rôznych typov výbušnín v digitálnej forme, kde bude mať používateľ vyhradený čas na ich zneškodnenie.

Bomba bude mať rôzne atribúty, ako napríklad jej typ, sériové číslo, výsledok zneškodnenia a ďalšie. Taktiež bude mať rozličné charakteristiky a metódy. Hráč dostane podľa výberu náročnosti určitý typ bomby. Príkladom môže byť výbušnina len s farebnými káblikmi a pomocou metódy Cut bude môcť účastník prestrihnúť kábel charakteristickej farby z výberu ponúknutých možností. Vyskytnú sa tam iné typy ako bomba s tlačidlami, kde sa využije metóda Push. Ďalší typ môže byť na PIN alebo na baterky. Pomocou dedenia hráč môže dostať aj viac typovú bombu, kde bude záležať na počte modulov. Čím viac modulov hráč dostane, tým bude čas na ich vyriešenie dlhší. Na konci ťahu, používateľ vždy dostane informáciu či zneškodnenie prebehlo úspešne alebo došlo ku simulovanej explózii. Ako náhle bomba vybuchne, hráč prehrá a musí hrať od začiatku.

Jednými z hlavných atribútov budú čas a počet pokusov. To znamená, že hráč bude mať príležitosť sa pomýliť a naďalej pokračovať v hre. Na zlepšenie hry budú pridané aj logické úlohy, ktoré majú naviesť používateľa na jej úspešné dokončenie. Pri PIN-ovom type bomby, bude mať hráč za úlohu adresný kód nejako zistiť. To bude môcť pomocou rotovania bomby alebo ho dostane priamo z konzoly. Budú sa tam vyskytovať aj kľúče a vypínače, ktoré budú mať za úlohu sprístupniť tlačidlá na zneškodnenie.

Hráč za úspešne vyriešenie bude mať možnosť dostať skóre so zapísaným časom odpočítavania. Všetky možnosti a úlohy budú generované vo vstupe a účastník si ich bude môcť vyberať pomocou im priradených čísel alebo znakov.

Usporiadanie balíkov



Main

```java
package sk.stuba.fiit.bombMaterials;

public class Main {

    public static void main(String[] args) {

        // the user is processing tasks of the bomb to defuse
        User user = new User();
        user.init();
    }
}
```

Rozhranie (Interface) – použitý iba v tomto balíku

```java
package sk.stuba.fiit.bomb;

// Interface

public interface Bomb {

    public void init();

}

```

Dedenie (Inheritance) – rodič (parent)

- Metóda použitá na overriding u podtried
- Final metóda bombDefused
- Polymorfizmus využitý vo všetkých podtriedach

```java
package sk.stuba.fiit.bomb;

import sk.stuba.fiit.levels.Levels;

public class Defusing {

    Levels levels = new Levels();

    public void init() {}

    // method use for overriding (prekonavanie) - protected
    protected void list() {
        System.out.println("\t- [yes]");
        System.out.println("\t- [no]");
    }

    // final
    // method use to announce explosion
    protected void explosion() {
        System.out.println("\n\t### Boom! ###");
        System.out.println("\n"
                + "      _.-^^---....,,--       \n"
                + "  _--                  --_  \n"
                + "<                        >)\n"
                + "|                         | \n"
                + " \\._                   _./  \n"
                + "    ```--. . , ; .--'''       \n"
                + "          | |   |             \n"
                + "       .-=||  | |=-.   \n"
                + "       `-=#$%&%$#=-'   \n"
                + "          | ;  :|     \n"
                + " _____.,-#%&$@%#&#~,._____");
        System.exit(0);
    }

    // method use to announce defusing of the bomb
    protected final void bombDefused() {
        System.out.println("Congrats! The bomb is defused.");
        levels.init();
    }

}
```

Využitý interface s dedením, agregácia, final atribúty

```java
package sk.stuba.fiit.bomb;

import java.util.ArrayList;

// this level will be always different because of changed colors of wires, which are always randomly generated

// Polymorphism
public class EVersionOne extends Defusing implements Bomb {


    // constants -> final
    final int numberOfWires = 6;
    final int timeToDefuse = 10;

    // aggregation
    Wire wire = new Wire();
    Health attempts = new Health();
    Print print = new Print();

    // static variable due to a variable tasks
    static List<String> colors = new ArrayList<String>();

    // overriding (prekonavanie)
    protected void list() {
        System.out.println("\tWhich wire are you going to cut?");

        for(int  i = 0; i < numberOfWires; i++) {
            System.out.println("\t- [" + colors.get(i)+ "]");
        }
    }

    /*
     * the most important method in this class
     * because there we are calling other methods and objects
     */

    @Override
    public void init() {
        // TODO Auto-generated method stub

        /* encapsulation (zapuzdrenie) -> set number
        of wires in the class wires to use its functions */
```

V MVersionOne a HVersionOne je využitý overloadnig

```java
    // overriding (prekonavanie)
    protected void list() {

        System.out.println("\tWhich wire are you going to cut?");
        for(int  i = 0; i < numberOfWires; i++) {
            System.out.println("\t- [" + colorsOfWires.get(i)+ "]");
        }
    }

    void listYesNo() {
        System.out.println("\t- [yes]");
        System.out.println("\t- [no]");
    }

    // overloading (pretazovanie)
    void list(String chars) {

        System.out.println("\tWhich button are you going to push " + chars + "?");
        for(int  i = 0; i < numberOfCoverButtons; i++) {
            System.out.println("\t- [" + colorsOfCoverButtons.get(i)+ "]");
        }
    }
```

User – využitie singletonu

```java
package sk.stuba.fiit.bombMaterials;

import java.util.Scanner;

public class User {

    // initialization of the object levels
    // aggregation
    Levels levels = new Levels();
    Print print = new Print();

    // singleton method
    public static void nameUserData() {

        UserData userData = UserData.getInstance();
        UserData userData2 = UserData.getInstance();

        if (userData.equals(userData2)) {
            System.out.println("names equals");
        } else {
            System.out.println("names don't equals");
        }
    }

    void init() {

        //class which is part of Java util used to receive user input
        Scanner input = new Scanner(System.in);

        System.out.println("Welcome to the game Defuse a bomb!");
        System.out.println("\n"
                + " . . .                              \n"
                + "              \\|/                      \n"
                + "            `--+--'                   \n"
                + "              /|\\                      \n"
                + "             ' | '                    \n"
                + "               |                      \n"
                + "               |                      \n"
                + "           ,--'#`--.                  \n"
                + "           |#######|                  \n"
                + "        _.-'#######`-._               \n"
                + "     ,-'###############`-.            \n"
```

```java
        System.out.println("Wanna be named? [yes/no]");
        //input
        String nameUser = input.nextLine();

        while(nameUser != null) {

            if (nameUser.equals("no")) {
                System.out.println("Okay, let's continue");
                break;
            } else if (nameUser.equals("yes")) {
                System.out.println("Named as John");
                nameUserData();
                break;
            } else {
                // call method to announce user wrong input
                print.incorrectInput();
                System.out.println("\n'Can we start? [yes/no]'");

                //input
                nameUser = input.nextLine();
            }

        }
        nameUser = null;
```

Singleton

```java
package sk.stuba.fiit.singleton;

// source: https://www.youtube.com/watch?v=s13rdEk5_xk

public class UserData {

    private static UserData instance = null;

     public String name;

    private UserData() {
        name = "John";
    }

    public static UserData getInstance() {

        if (instance == null) {

            instance = new UserData();
        }

        return instance;
    }
}
```

Zapuzdrenie (Encapsulation) v triede Wire

- Použitie Random z java.util

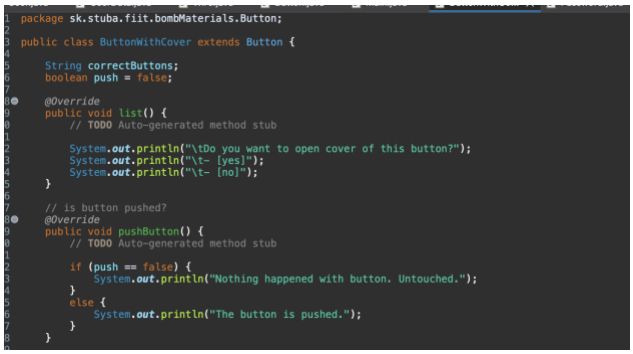
```java
    // Encapsulation (zapuzdrenie)
    // get list of colors
    public String getRandomColorsList(int i) {
        return randomColors.get(i);
    }

    // get variable of numberW
    public int getNumberOfWires() {
        return numberOfWires;
    }

    // set variable of numberW
    public void setNumberOfWires(int newNumberOfWires) {
        numberOfWires = newNumberOfWires;
    }

}
```

```java
    List<String> randomColors = new ArrayList<String>();

    // from the list of colors will be generated another list of colors but with a concrete number
    public List<String> generateRandomColors(List<String> colorsOfWires) {

        Collections.shuffle(colors);

        for(int  i = 0; i < numberOfWires; i++) {
            randomColors.add(colors.get(i));
        }

        for(int i = 0; i < numberOfWires; i++) {
            colorsOfWires.add(getRandomColorsList(i));
        }

        return colorsOfWires;
    }
```

Abstraktná trieda s abstraktnými metódami

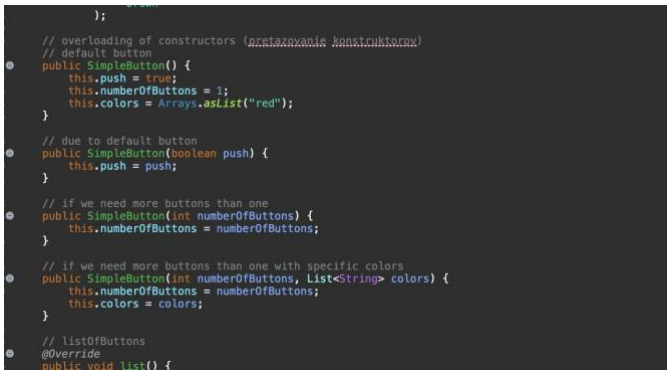
```java
package sk.stuba.fiit.bombMaterials.Button;

// an abstract class with abstract methods

public abstract class Button {

    public abstract void list();

    public abstract void pushButton();

    public abstract void checkButton(boolean check);

    public void pushButton(String doubleButtons) {
        // TODO Auto-generated method stub

    }

    public abstract boolean getPush();

    public abstract void setPush(boolean newPush);

    public void pushButton(String doubleButtons, String correctButtons) {
        // TODO Auto-generated method stub

    }

}
```

Príklad dedenia abstraktnej triedy



```java
package sk.stuba.fiit.bombMaterials.Button;

public class ButtonWithCover extends Button {

    String correctButtons;
    boolean push = false;

    @Override
    public void list() {
        // TODO Auto-generated method stub

        System.out.println("\tDo you want to open cover of this button?");
        System.out.println("\t- [yes]");
        System.out.println("\t- [no]");
    }

    // is button pushed?
    @Override
    public void pushButton() {
        // TODO Auto-generated method stub

        if (push == false) {
            System.out.println("Nothing happened with button. Untouched.");
        }
        else {
            System.out.println("The button is pushed.");
        }
    }
```

Preťažovanie konštruktorov - SimpleButton



```java
            );

    // overloading of constructors (pretazovanie konstruktorov)
    // default button
    public SimpleButton() {
        this.push = true;
        this.numberOfButtons = 1;
        this.colors = Arrays.asList("red");
    }

    // due to default button
    public SimpleButton(boolean push) {
        this.push = push;
    }

    // if we need more buttons than one
    public SimpleButton(int numberOfButtons) {
        this.numberOfButtons = numberOfButtons;
    }

    // if we need more buttons than one with specific colors
    public SimpleButton(int numberOfButtons, List<String> colors) {
        this.numberOfButtons = numberOfButtons;
        this.colors = colors;
    }

    // listOfButtons
    @Override
    public void list() {
```

```java
    // Encapsulation (zapuzdrenie)
    // get list of colors
    public String getRandomColorsList(int i) {
        return randomColors.get(i);
    }

    // Encapsulation (zapuzdrenie)
    // get variable push
    @Override
    public boolean getPush() {
        // TODO Auto-generated method stub

        return push;
    }

    // set variable push
    @Override
    public void setPush(boolean newPush) {
        // TODO Auto-generated method stub

        push = newPush;
    }

}
```

Použitie super a asociácia

```java
public class Health extends Defusing {

    int healthAttempts = 3;

    // super
    // overriding (prekonavanie)
    protected void explosion() {
        System.out.println("\n<<< You've lost all of your attempts. :( >>>");
        super.explosion();
    }
```

```java
// association (asociacia)
// check how many attempts the user has
public void lifeLoss(Countdown countdown) {
    // if more than one live
    if (healthAttempts > 1) {
        System.out.println("<<< You have only " + healthAttempts + " attempts from 3. >>>");
    // else if only one live
    } else if ((healthAttempts <= 1) && (healthAttempts > 0)) {
        System.out.println("<<< You have only " + healthAttempts + " attempt from 3. >>>");
    // else boom
    } else {
        countdown.stop();
        explosion();
    }
}
```

Použitie downcastingu a upcastingu v triede Easy

```java
void init() {

    number = generateRandomELevel();

    EVersionOne v1 = new EVersionOne();
    EVersionTwo v2 = new EVersionTwo();
    EVersionThree v3 = new EVersionThree();

    // upcasting na Defusing
    Defusing def1 = v1;
    Defusing def2 = v2;
    Defusing def3 = v3;

    EVersionOne v1_2 = null;
    EVersionTwo v2_2 = null;
    EVersionThree v3_2 = null;

    // downcasting
    switch(number) {

        case n1:

            if (def1 instanceof EVersionOne) {
                v1_2 = (EVersionOne) def1;
                v1_2.init();
            }
            break;

        case n2:

            if (def2 instanceof EVersionTwo) {
                v2_2 = (EVersionTwo) def2;
                v2_2.init();
            }
            break;

        case n3:

            if (def3 instanceof EVersionThree) {
                v3_2 = (EVersionThree) def3;
```

Kompozícia

```java
    String pickLevel = input.nextLine();

    while(pickLevel != null) {

        // if 'easy'
        if (pickLevel.equals(listOfLevels.get(0))) {
            // composition (kompozicia)
            // call constructor Easy
            Easy easy = new Easy();
            easy.init();
            input.close();
            return;
        // if 'medium'
        } else if (pickLevel.equals(listOfLevels.get(1))) {
            // composition (kompozicia)
            Medium medium = new Medium();
            medium.init();
            input.close();
            return;
        // if 'hard'
        } else if (pickLevel.equals(listOfLevels.get(2))) {
            // composition (kompozicia)
            Hard hard = new Hard();
            hard.init();
            input.close();
            return;
        // wrong input
        } else if (pickLevel.equals(listOfLevels.get(3))) {
            System.out.println("Quit");
            System.exit(0);
        } else {
            print.incorrectInput();
            printLevels();

            //input
            pickLevel = input.nextLine();
        }
```

Statická premenná

```java
static List<String> switches =
                    Arrays.asList(
                        "switch 1",
                        "switch 2",
                        "switch 3",
                        "switch 4"
                    );
```

Statická metóda → použitá ako Switch.(metóda)

```java
public static void turnOnSwitch() {
    System.out.println("\nThe Switch is turned on.");
}

public static void turnOffSwitch() {
    System.out.println("The Switch is turned off.");
}
```