Vadym Popovych

Cvičenie: -2.01(CPUa), streda 12:00, Ing. Abd Alrahman Saleh

Projekt z objektovo orientovaného programovania LS2025

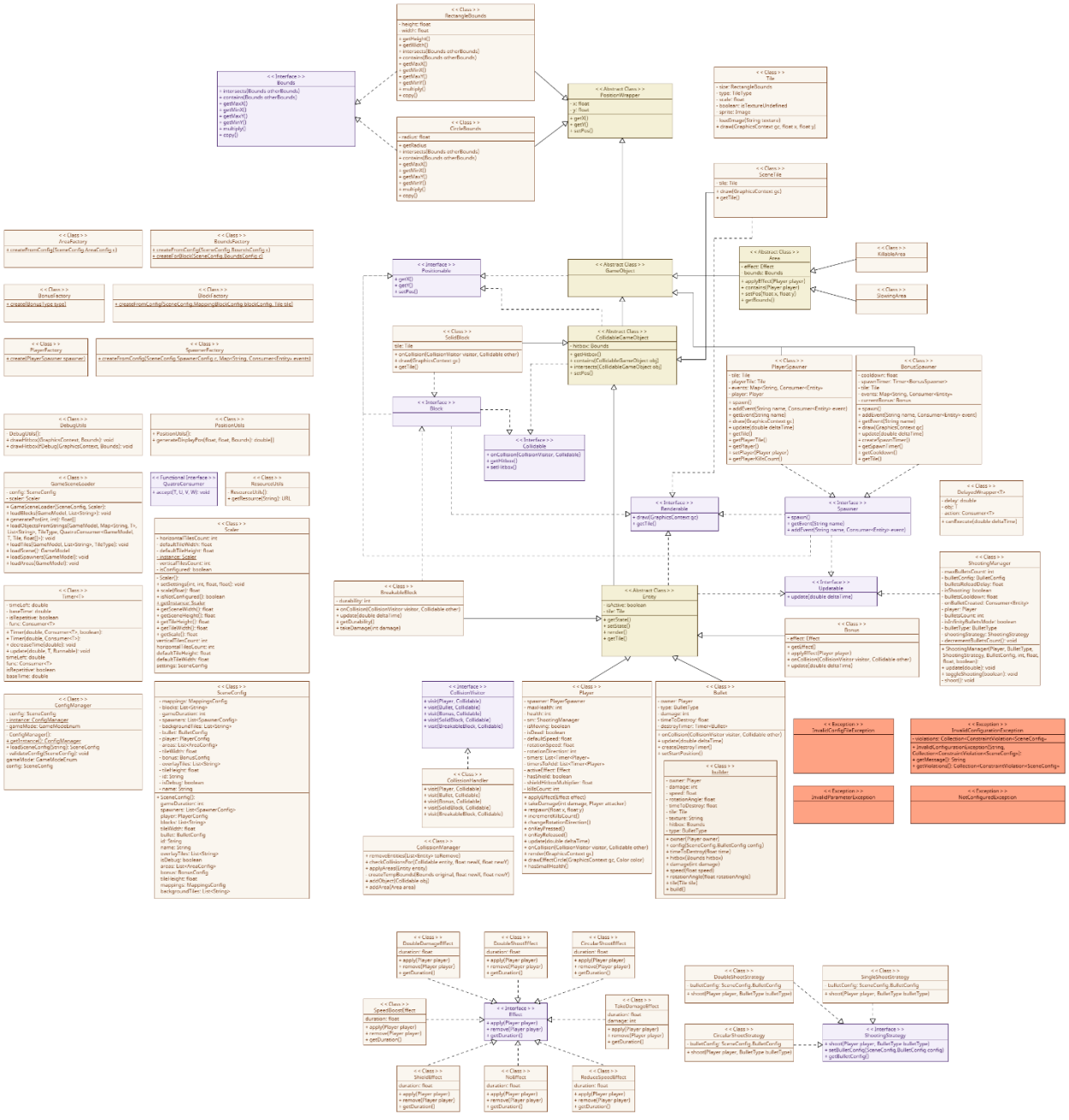
Zámer projektu

"Multiplayer Shooter minigame" is a dynamic 2D shooter for two players with a fixed camera. The characters constantly rotate, and when their controll key is pressed, they start moving in the chosen direction and shooting. The map contains static objects that block passage, shelters that can be passed through, and various power-ups. Players can collect temporary bonuses such as a shield, increased movement speed, faster shooting, or enhanced damage.

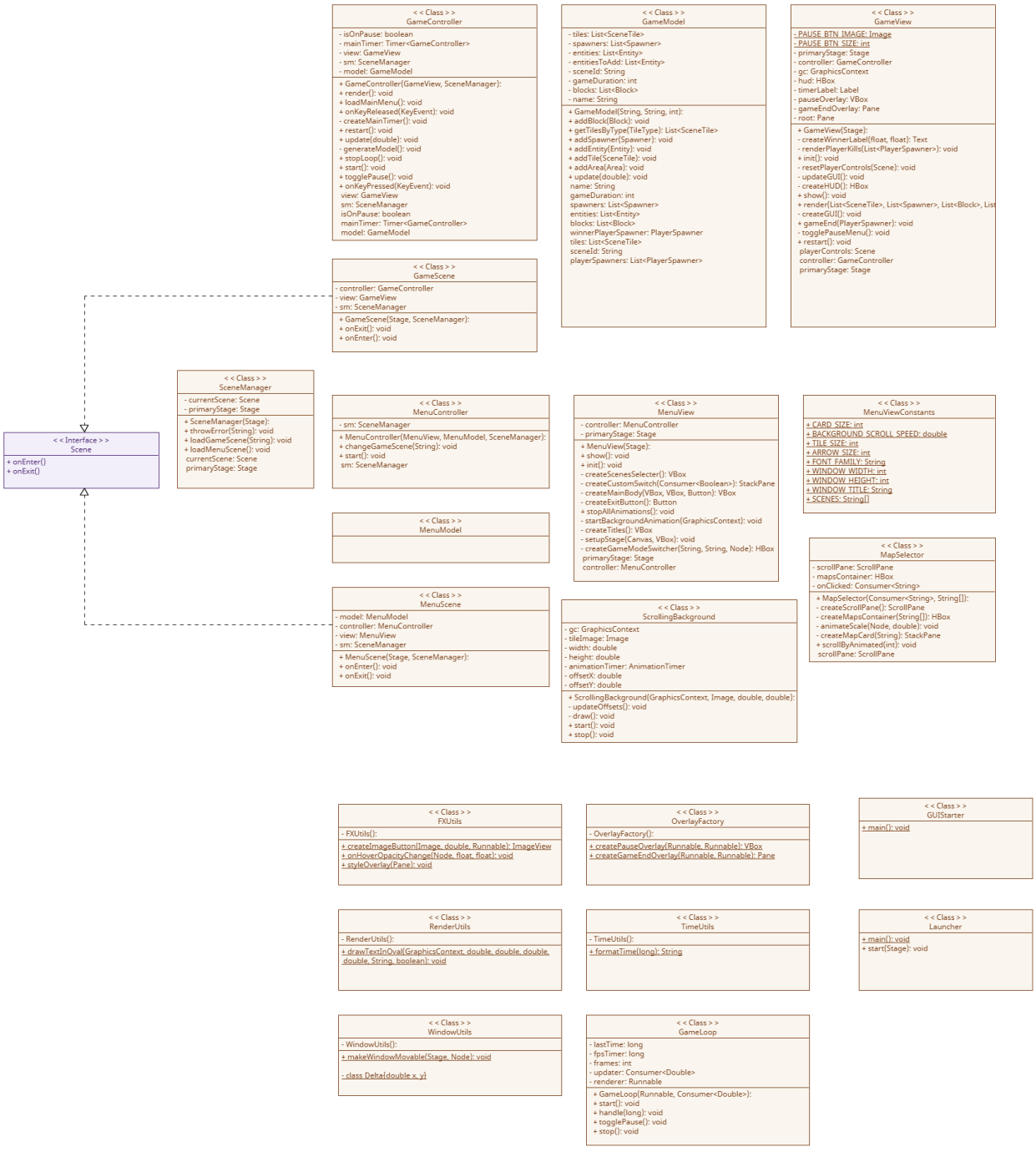
The game has two modes: one with unlimited ammo and another with a limited number of bullets that gradually regenerate. In most cases, a character dies after two hits — after the first shot, they visibly turn red. Power-up effects change the appearance of projectiles or the character itself. Upon death, the player respawns at a special location and gains temporary immunity. The match lasts for a set amount of time, and the winner is the player who destroys their opponent more times.

Genre: arcade shooter

UML diagram tried



If you notice that the image quality is very low, there are files named uml_core.pdf and uml_gui.pdf in the docs/ folder, and you can view them in the highest quality.


< < Class > >
GameController
- isOnPause: boolean
- mainTimer: Timer<GameController>
- view: GameView
- sm: SceneManager
- model: GameModel
+ GameController(GameView, SceneManager):
+ render(): void
+ loadMainMenu(): void
+ onKeyReleased(KeyEvent): void
- createMainTimer(): void
+ restart(): void
+ update(double): void
- generateModel(): void
+ stopLoop(): void
+ start(): void
+ togglePause(): void
+ onKeyPressed(KeyEvent): void
view: GameView
sm: SceneManager
isOnPause: boolean
mainTimer: Timer<GameController>
model: GameModel
< < Class > >
GameModel
- tiles: List<SceneTile>
- spawners: List<Spawner>
- entities: List<Entity>
- entitiesToAdd: List<Entity>
- sceneId: String
- gameDuration: int
- blocks: List<Block>
- name: String
+ GameModel(String, String, int):
+ addBlock(Block): void
+ getTilesByType(TileType): List<SceneTile>
+ addSpawner(Spawner): void
+ addEntity(Entity): void
+ addTile(SceneTile): void
+ addArea(Area): void
+ update(double): void
name: String
gameDuration: int
spawners: List<Spawner>
entities: List<Entity>
blocks: List<Block>
winnerPlayerSpawner: PlayerSpawner
tiles: List<SceneTile>
sceneId: String
playerSpawners: List<PlayerSpawner>
< < Class > >
GameView
- PAUSE_BTN_IMAGE: Image
- PAUSE_BTN_SIZE: int
- primaryStage: Stage
- controller: GameController
- gc: GraphicsContext
- hud: HBox
- timerLabel: Label
- pauseOverlay: VBox
- gameEndOverlay: Pane
- root: Pane
+ GameView(Stage):
- createWinnerLabel(float, float): Text
- renderPlayerKills(List<PlayerSpawner>): void
+ init(): void
- resetPlayerControls(Scene): void
- updateGUI(): void
- createHUD(): HBox
+ show(): void
+ render(List<SceneTile>, List<Spawner>, List<Block>, List
- createGUI(): void
+ gameEnd(PlayerSpawner): void
- togglePauseMenu(): void
+ restart(): void
playerControls: Scene
controller: GameController
primaryStage: Stage
< < Class > >
GameScene
- controller: GameController
- view: GameView
- sm: SceneManager
+ GameScene(Stage, SceneManager):
+ onExit(): void
+ onEnter(): void
< < Interface > >
Scene
+ onEnter()
+ onExit()
< < Class > >
SceneManager
- currentScene: Scene
- primaryStage: Stage
+ SceneManager(Stage):
+ throwError(String): void
+ loadGameScene(String): void
+ loadMenuScene(): void
currentScene: Scene
primaryStage: Stage
< < Class > >
MenuController
- sm: SceneManager
+ MenuController(MenuView, MenuModel, SceneManager):
+ changeGameScene(String): void
+ start(): void
sm: SceneManager
< < Class > >
MenuView
- controller: MenuController
- primaryStage: Stage
+ MenuView(Stage):
+ show(): void
+ init(): void
- createScenesSelecter(): VBox
- createCustomSwitch(Consumer<Boolean>): StackPane
- createMainBody(VBox, VBox, Button): VBox
- createExitButton(): Button
+ stopAllAnimations(): void
- startBackgroundAnimation(GraphicsContext): void
- createTitles(): VBox
- setupStage(Canvas, VBox): void
- createGameModeSwitcher(String, String, Node): HBox
primaryStage: Stage
controller: MenuController
< < Class > >
MenuViewConstants
+ CARD_SIZE: int
+ BACKGROUND_SCROLL_SPEED: double
+ TILE_SIZE: int
+ ARROW_SIZE: int
+ FONT_FAMILY: String
+ WINDOW_WIDTH: int
+ WINDOW_HEIGHT: int
+ WINDOW_TITLE: String
+ SCENES: String[]
< < Class > >
MenuModel
< < Class > >
MenuScene
- model: MenuModel
- controller: MenuController
- view: MenuView
- sm: SceneManager
+ MenuScene(Stage, SceneManager):
+ onEnter(): void
+ onExit(): void
< < Class > >
ScrollingBackground
- gc: GraphicsContext
- tileImage: Image
- width: double
- height: double
- animationTimer: AnimationTimer
- offsetX: double
- offsetY: double
+ ScrollingBackground(GraphicsContext, Image, double, double):
- updateOffsets(): void
- draw(): void
+ start(): void
+ stop(): void
< < Class > >
MapSelector
- scrollPane: ScrollPane
- mapsContainer: HBox
- onClicked: Consumer<String>
+ MapSelector(Consumer<String>, String[]):
- createScrollPane(): ScrollPane
- createMapsContainer(String[]): HBox
- animateScale(Node, double): void
- createMapCard(String): StackPane
+ scrollByAnimated(int): void
scrollPane: ScrollPane
< < Class > >
FXUtils
- FXUtils():
+ createImageButton(Image, double, Runnable): ImageView
+ onHoverOpacityChange(Node, float, float): void
+ styleOverlay(Pane): void
< < Class > >
OverlayFactory
- OverlayFactory():
+ createPauseOverlay(Runnable, Runnable): VBox
+ createGameEndOverlay(Runnable, Runnable): Pane
< < Class > >
GUIStarter
+ main(): void
< < Class > >
RenderUtils
- RenderUtils():
+ drawTextInOval(GraphicsContext, double, double, double, double, String, boolean): void
< < Class > >
TimeUtils
- TimeUtils():
+ formatTime(long): String
< < Class > >
Launcher
+ main(): void
+ start(Stage): void
< < Class > >
WindowUtils
- WindowUtils():
+ makeWindowMovable(Stage, Node): void
- class Delta{double x, y}
< < Class > >
GameLoop
- lastTime: long
- fpsTimer: long
- frames: int
- updater: Consumer<Double>
- renderer: Runnable
+ GameLoop(Runnable, Consumer<Double>):
+ start(): void
+ handle(long): void
+ togglePause(): void
+ stop(): void

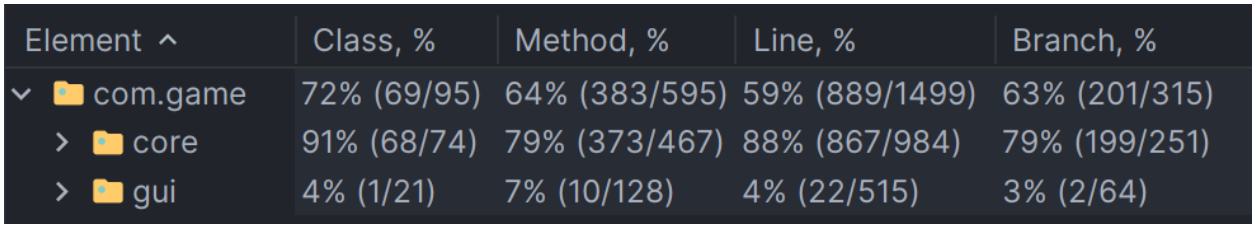
Použité knižnice

1. *org.openjfx:javafx-controls, javafx-graphics – used for GUI*
2. *com.fasterxml.jackson.core:jackson-databind – used for loading configs*
3. *com.fasterxml.jackson.core: jackson-dataformat-yaml – used for reading yaml logging config*
4. *org.hibernate.validator:hibernate-validator, org.glassfish.expressly:expressly – used for config validation*
5. *org.apache.logging.log4j:log4j-api, log4j-core, log4j-slf4j2-impl, org.slf4j:slf4j-api – used for logging, slf4j2 used as facade of logging*
6. *junit:junit, org.mockito:mockito-core, mockito-junit-jupiter – used for testing and mocking classess*

Vyjadrenie sa k splneniu podmienok (nutnych, aj dalsich)

Ku kazdej podmienke napiste aspon jeden priklad kde v projekte je splnena, dodrzujte prosim poradie ako je na uvedene na stranke projekt:

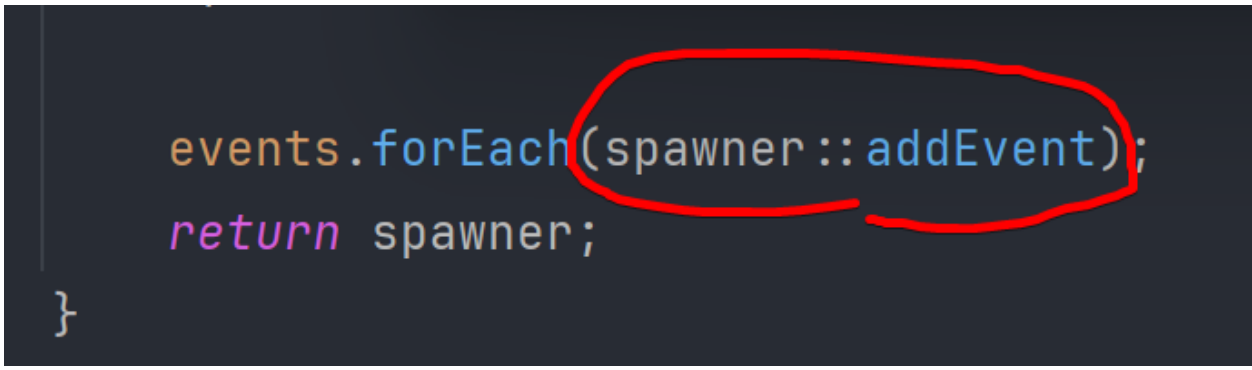
- *Obsahovat dedenie a rozhrania:*
    - *Class Area extends from GameObject*
    - *Interface Block extends from Collidable, Renderable, Positionable*
    - *Class CollidableGameObject extends from GameObject*
    - *Class Entity extends from CollidableGameObject*
    - *Classes Player, Bullet, Bonus extends from Entity*
    - *Class BreakableBlock extends from Entity*
    - *There are four main interfaces: Collidable, Updatable, Renderable, Positionable (core.behaviour.interfaces)*
    - *Also, there are other interfaces: Effect (core.effects), ShootingStrategy (core.strategies), Block (core.scene.blocks), Spawner (core.scene.spawners)*
- *V pripade pouzitia inych externych zavislosti musi byt pouzity Maven*
    - *As you can see in project files there is file pom.xml(configuration for maven)*
- *Pouzitie zakladnych OOP principov (enkapsulacia, dedenie, polymorfizmus, abstrakcia)*
    - *Encapsulation: in all classes used private modifier to class variables, for example class Player (core.entities)*

    - *Inheritance: already described in the top of the list*
    - *Polymorphism: Interface Effect implementing by all effects, interface Renderable implementing by objects that must be rendered in the game window(Player, Bullet, Bonus, SolidBlock...)*
    - *Abstraction: used such abstract classes as Area (core.scene.areas), PositionWrapper (core.behaviour.base), Entity (core.entities)*
- *Musi obsahovat dostatok komentorov a anotacii (JavaDoc)*
    - *For all classes and almost all methods, except simple getters/setters, Javadoc is written.*
- *Musi byt jednotkovo otestovany*
    - *Tested all `core` package for over than 85% of line coverage*
    - 

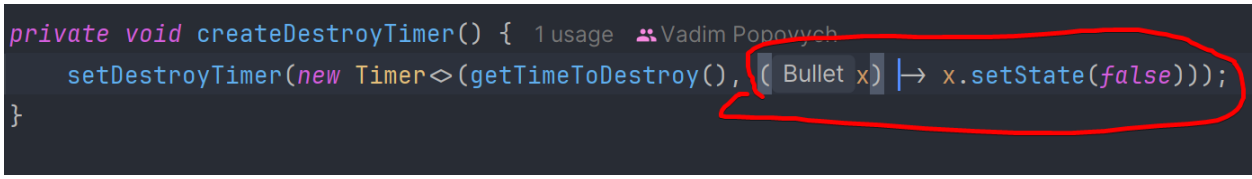
Element	Class, %	Method, %	Line, %	Branch, %
com.game	72% (69/95)	64% (383/595)	59% (889/1499)	63% (201/315)
core	91% (68/74)	79% (373/467)	88% (867/984)	79% (199/251)
gui	4% (1/21)	7% (10/128)	4% (22/515)	3% (2/64)

- *Pouzitie navrhovych/architektonickych vzorov*
    - *Used Factory pattern in core.factories package*
    - *Used Visitor pattern in core.collisions.CollisionHandler. It is used to avoid the accumulation of "if" statements with "instance of"*
    - *Used Strategy pattern in core.strategies*
    - *Used Builder pattern in core.entities.bullet.Bullet for convenient Bullet creation*
    - *Used Singleton pattern in core.utils.config.ConfigManager and in core.utils.Scaler*
- *Logovanie zakladnych cinnosti*
    - *Used logging in almost all `core` classes, for example logging in Player(line 113, 120, 125, ...), CollisionManager(line 59), Tile(line 69, 92, 98, ...)*
    - *Logging config file is placed at resources/log4j2.yaml*
- *Implementacia a pouzitie vlastnych vynimiek*
    - *In package core.exceptions are placed four custom exceptions – InvalidConfigFileException, InvalidConfigurationException, InvalidParameterException, NotConfiguredException*
    - *InvalidConfigFileException – used in case when provided invalid scene config file(core.utils.config.ConfigManager.loadSceneConfig)*
    - *InvalidConfigurationException – throwed when during config validation are found violations(core.utils.config.ConfigManager.validateConfig)*

    - *InvalidParameterException – used in many methods to throw when provided parameter is invalid for any reason(core.entities.Player.setMaxHealth, core.behaviour.bounds.RectangleBounds.multiply, core.behaviour.bounds.RectangleBounds.setWidth, ...)*
    - *NotConfiguredException – throwed when in object not set specific config(core.utils.Scaler.getTileWidth, core.utils.Scaler.getTileHeight, core.entities.Entity.move)*
- *Implementacia a vytvorenie GUI*
    - *GUI is created with JavaFx, code is placed in `gui` package*
- *Pouzitie generickovsti vo vlasnych triedach*
    - *Used in core.utils.Timer and in core.shooting.DelayedWrapper*
- *Pouzitie reflexive*
    - *Used in tests for accessing private fields(TileTest. constructorWithNonDefaultSizeUsesSpriteSizes, EntityTest. setupConfig, ...)*
- *Pouzitie lambda vyrazov, referencii na metody*
    - *In core.utils.GameSceneLoader - loadTiles, loadBlocks*
    - *core.factories.SpawnerFactory, line 50*



    - *core.entities.Bullet*



Navod ako spustit projekt

- Start point of the application is class GUIStarter, this class you should run if you want to start application

Pouzivatelska prirucka

- Image on the center of window is the game map, to start game you should click on this image. By clicking on arrows around image you can select another maps. yThe

switcher at the top of this brown board is responsible for switching between the game modes. On the bottom of the window, you can see `Exit` button.

- The red arrows point to the players. The green arrow indicates the timer, and the blue arrows point to the bonus spawners. Additionally, in the top-right corner, there is a button that opens the pause menu. Gray circles with number near spawners is showing number of player kills (On the image is presented map 2)