**Zadanie 3: Použité OOP Princípy**

Ondrej Bodnár, B-INFO4

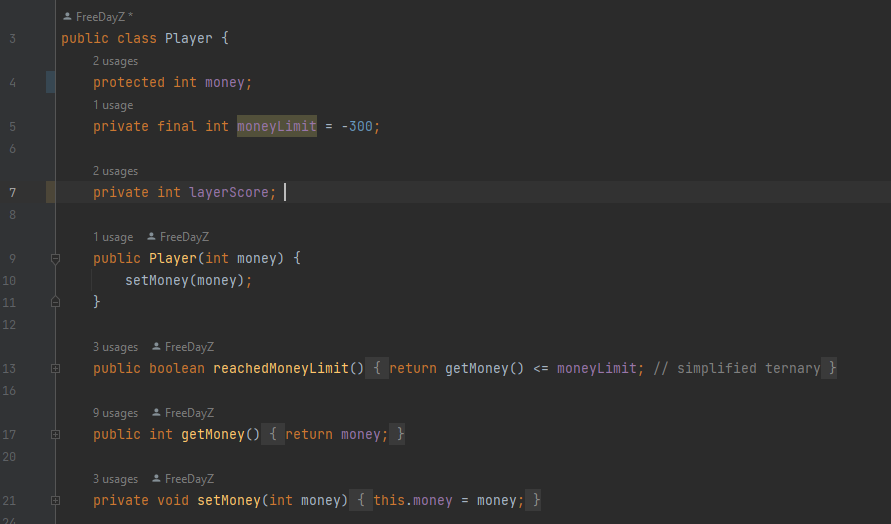
1. **Dedenie**

V projekte som využil princípy dedenia napríklad pri rozšírení triedy Marketplace triedou HappyHourMarket. Vďaka tomuto rozšíreniu hráč získa náhodný bonus pri predávaní svojich resources. V tomto prípade som využil aj override pôvodnej metódy v parent triede Marketplace.



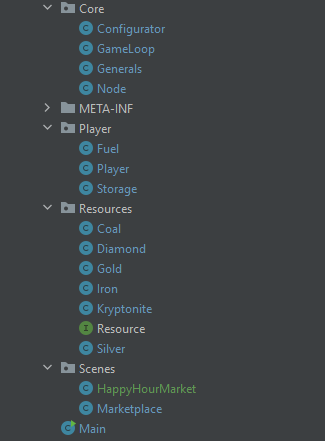
1. **Zapuzdrenie**

Princíp zapuzdrenia som využil v celom kóde programu. Napríklad pri triede Player, kde atribút money predstavuje senzitívnu informáciu, ktorú som chcel zabezpečiť princípom zapuzdrenia. Využitím getter a setter metód som zabezpečil, aby nastavovanie peňazí daného hráča bolo možné iba v danej triede (s pomocou private modifikátora metódy setMoney()). Následne môzme využívať getter metódu kdekoľvek v programe, ktorá nám vráti množstvo peňazí, ktorými daný hráč disponuje.



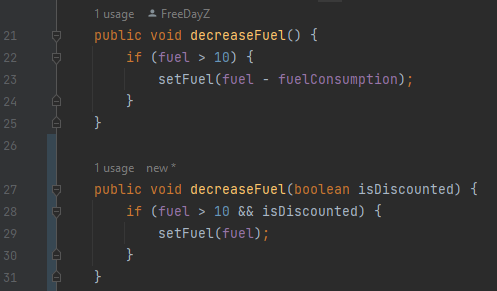
1. **Organizácia kódu**

Program je organizovaný do viacerých balíkov následovne:



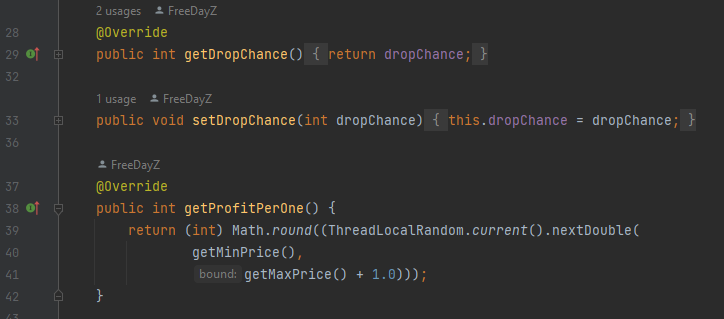
1. **Overloading (preťaženie) metód**

Preťaženie metód som využil napríklad pri odoberaní paliva hráčovi. Postupom hry má hráč 5% šancu na získanie “zľavy” paliva. To znamená, že následujúci layer(podlažie) mu nebude odrátaná spotreba paliva (10litrov).



1. **Overriding (prekonanie) metód**

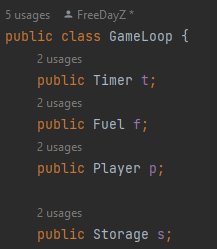
Prekonávanie metód som využíval vo viacerých prípadoch. Každá trieda má prekonané metódy toString(), equals() a hashCode(). Prekonávanie som využil tiež pri nastavovaní behu programu (TimerTask), kde bolo nutné implementovať vlastný TimerTask s prekonaním default metódy run(). Okrem toho som využil prekonávanie pri vytváraní tried s implementáciou rozhrania Resource.





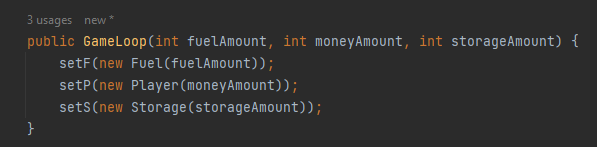
1. **Agregácia**

Agregáciu som využil v hlavnom cykle programu. Zabezpečíme vzťah medzi triedou GameLoop a triedami Timer,Fuel,Player,Storage pomocou atribútov a enkapsulácie. Inštancie daných tried sú vytvorené v triede GameLoop iba raz.



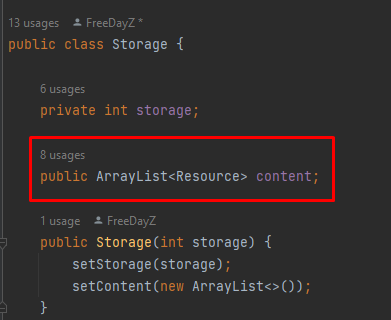
1. **Kompozícia**

Princíp kompozície bol využitý najmä v hlavnom cykle programu. Pomocou getter a setter metód som si vytvoril inštancie, ktoré sú vytvárané jedine v tomto konštruktore, nikde inde v programe. V programe k ním pristupujem pomocou getter metódy.



**10. Asociácia**

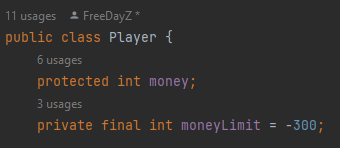
Vzťah pomocou asociácie je využitý napríklad v triede Storage, kde som použil ArrayList Resource rozhraní, vďaka čomu dokážem do atribútu content vložiť rôzne objekty implementujúce Resource rozhranie a to nám slúži ako “inventár” vyťažených resources.



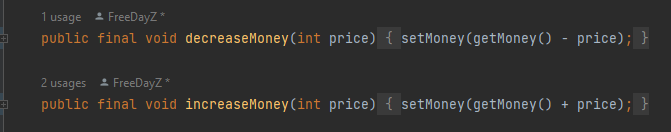
**11. Finálny atribút, finálna metóda**

Pomocou finálneho atribútu som zabezpečil to, aby nebolo možné zmeniť obsah tohto atribútu, pomocou ktorého je kód bezpečnejší. Taktiež som využil finálne metódy, ktoré po nastavení final nemôžu byť neskôr prekonané.

*Príklad finálneho atribútu moneyLimit*

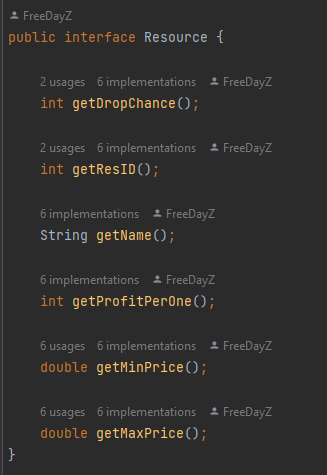


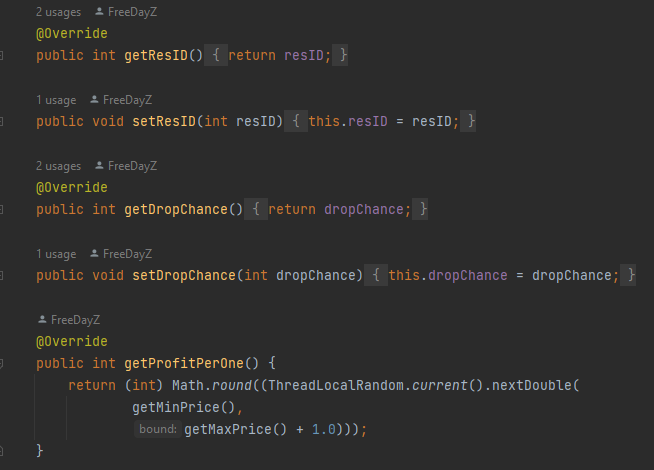
*Príklad finálnych metód*



**12. Rozhranie, abstraktná trieda**

V programe som využil rozhranie Resource. Toto rozhranie uľahčuje vytváranie nových resources, ktoré môže hráč počas hry objaviť. Triedy, ktoré implementujú dané rozhranie majú prekonané metódy v danom rozhraní.

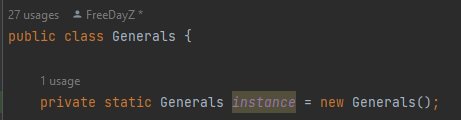




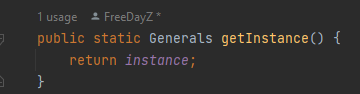
*Ukážka triedy Coal, ktorá implementuje rozhranie Resource*

**13. Statická metóda, statický atribút, Singleton**

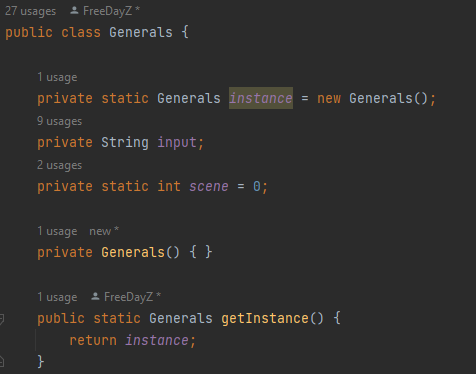
V programe som využil Singleton návrhový vzor pri triede Generals. Dosiahol som ho pomocou statického atribútu inštancie triedy Generals a statickej metódy getInstance(). Okrem toho je využitý privátny konštruktor. Pri vytváraní vo funkcii Main dokážeme získať inštanciu danej triedy bez jej vytvárania. Inštancia je vytvorená už v triede a my k nej pristupujeme pomocou statickej getInstance() metódy.



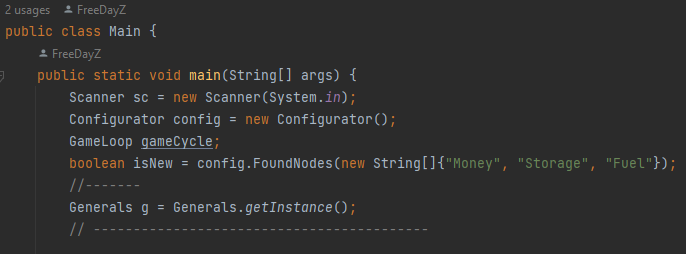
*Statický atribút instance*



*Statická metóda getInstance()*

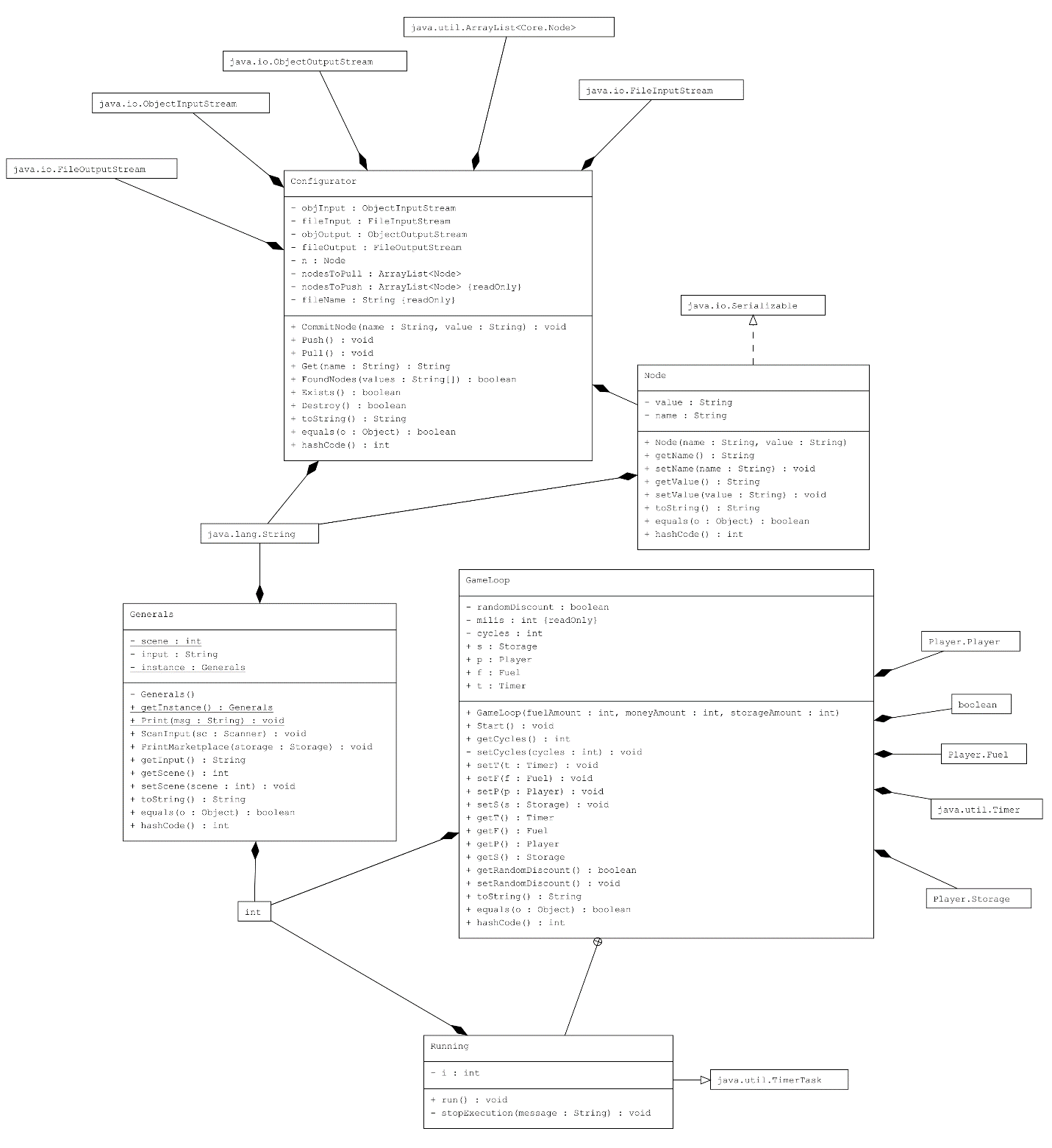


*Využitie návrhového vzoru Singleton v triede Generals*

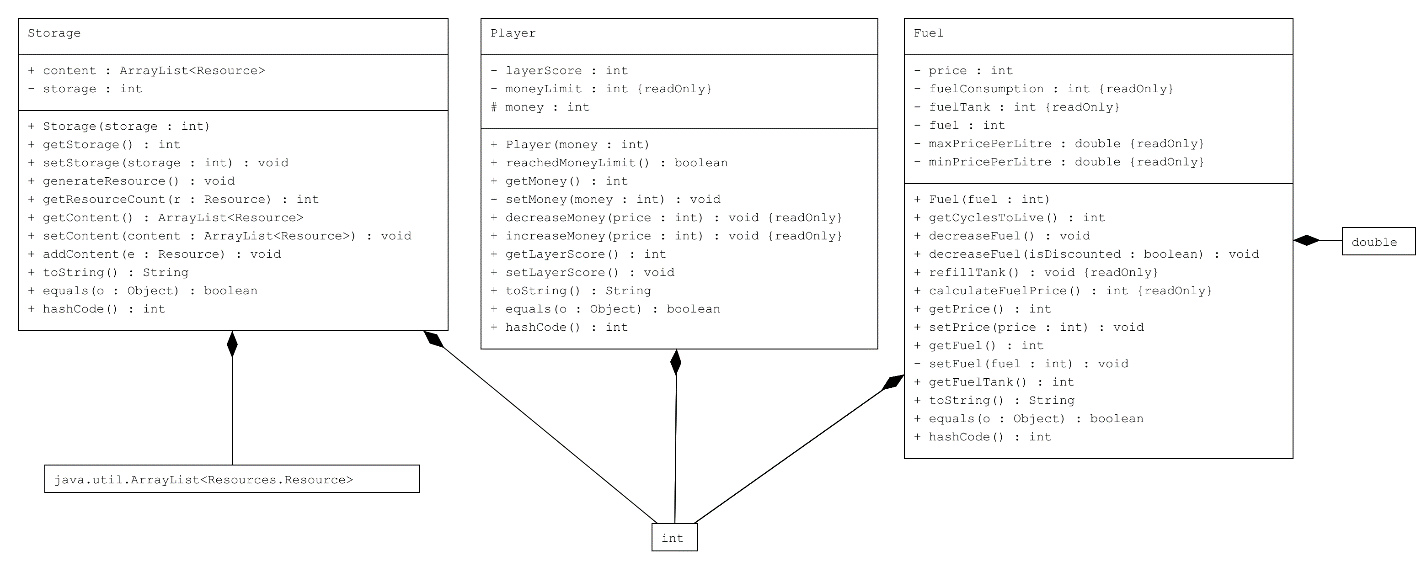


*Ukážka získania inštancie pomocou návrhového vzoru Singleton v Main.java*

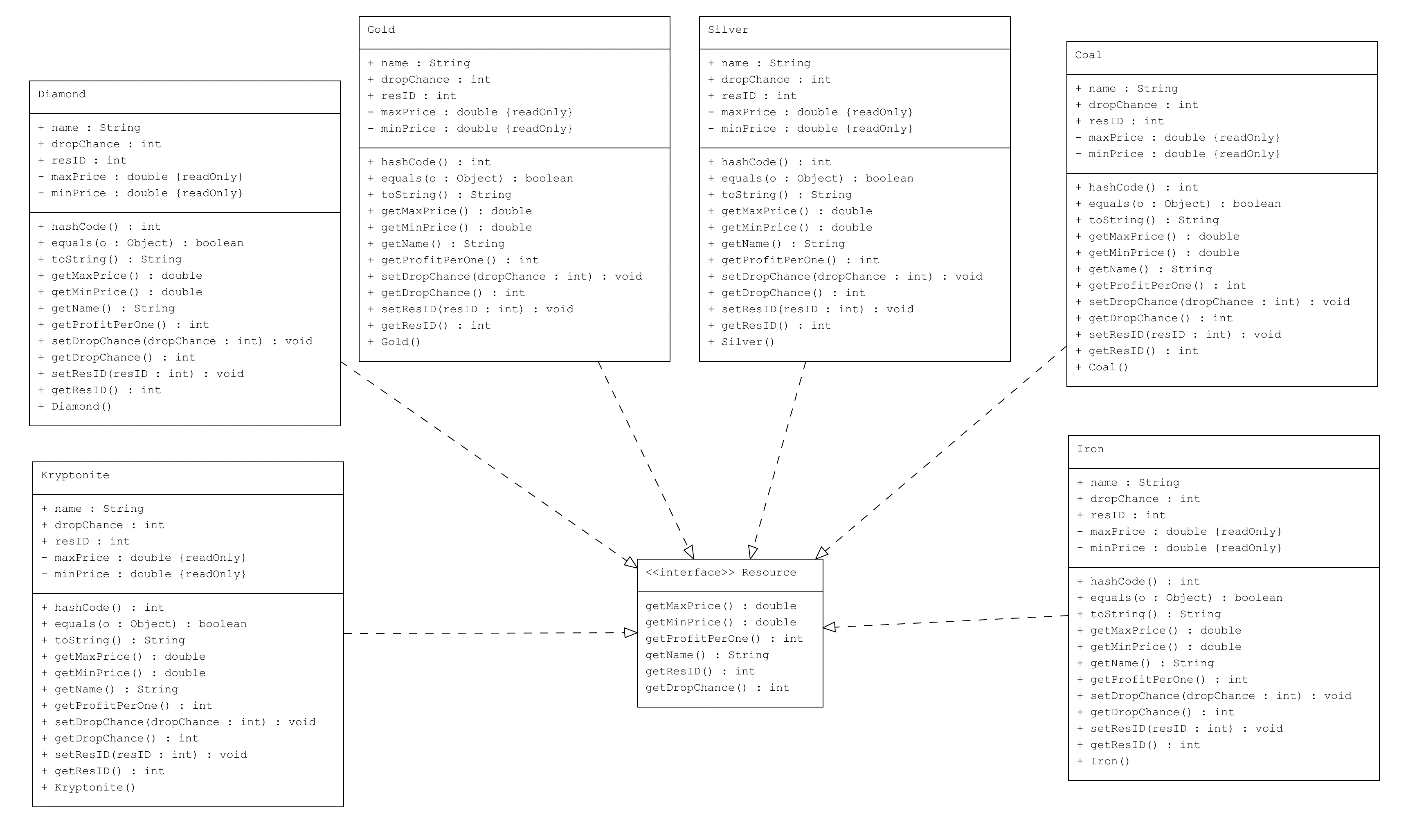
**UML Diagramy tried programu**

****

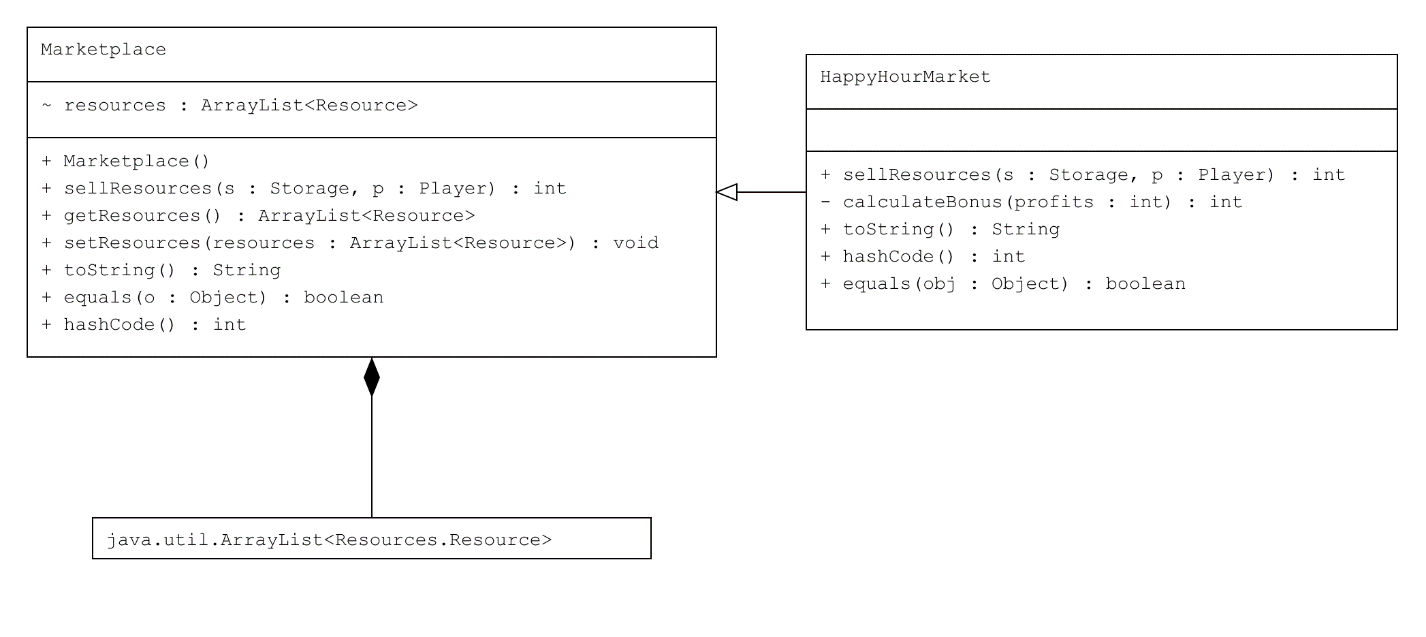
*Package Core*

**

*Package Player*

**

*Package Resources*

**

*Package Scenes*