Projekat 5 - GUI

Kreirati hijerarhiju klasa koja predstavlja jednostavnu igru sa igračem i neprijateljima, gdje se informacije o igraču (Player) unose putem grafičkog interfejsa (GUI), dok se neprijatelji (Enemy) učitavaju iz spoljašnjeg CSV fajla. Osnovu sistema čini apstraktna klasa **GameObject**, iz koje se izvode klase Player i Enemy. Klasa GameObject sadrži osnovnu logiku za poziciju na ekranu i koliziju, dok se konkretna geometrija objekta (pravougaonik ili krug) čuva preko posebnog interfejsa. Pozicija objekta se opisuje koordinatama x i y (tipa int), a oblik preko interfejsa Collidable. U klasi GameObject potrebno je definisati privatne atribute za poziciju i kolajder, konstruktor koji postavlja vrijednosti, odgovarajuće gettere i settere, kao i provjere validnosati. Kod nelogičnih vrijednosti (npr. negativnih dimenzija) baciti izuzetak IllegalArgumentException. Klasa treba da bude apstraktna i sadrži makar jednu apstraktnu metodu, kao što je public abstract String getDisplayName(). U okviru nje implementirati i metodu boolean intersects(GameObject other), koja provjerava koliziju delegiranjem poziva na kolajdere. Metoda toString() može davati generički opis objekta.

Interfejs **Collidable** predstavlja bilo koji objekat koji može da učestvuje u koliziji. Potrebno je definisati metodu boolean intersects(Collidable other). Na osnovu interfejsa treba napraviti dvije konkretne klase: **RectangleCollider** i **CircleCollider**. RectangleCollider sadrži poziciju, širinu i visinu, i u konstruktoru provjerava da su dimenzije veće od nule. Metoda intersects treba da omogućava koliziju sa drugim pravougaonikom, a ako je u pitanju krug, logika se prebacuje na krug. CircleCollider opisuje krug pomoću centra i poluprečnika, koji mora biti strogo pozitivan. Kod kolizije između kruga i pravougaonika koristi se standardan pristup, određuje se najbliža tačka pravougaonika centru kruga, i provjerava da li je udaljenost manja ili jednaka poluprečniku. Može se koristiti pomoćna metoda clamp za ograničavanje vrijednosti.

Iz GameObject klase nasljeđuju klase **Player** i **Enemy**. U konstruktorima ovih klasa, pored pozicije, prosleđuje se i odgovarajući kolajder (pravougaoni ili kružni). Klasa **Player** sadrži dodatne atribute name (String) i health (int od 0 do 100). Ime započinje velikim slovom. Ako nakon obrade ostane prazno, baca se izuzetak. Health se mora nalaziti u validnom opsegu. Metoda toString() vraća detaljan prikaz igrača, dok getDisplayName() vraća njegovo ime.

Klasa **Enemy** takođe nasljeđuje GameObject i implementira interfejs **Attacker**, koji sadrži metodu int getEffectiveDamage(). Pored osnovnih atributa pozicije i kolajdera, Enemy ima type (String), damage i health (int, oba u opsegu 0 do 100). Damage i health se validiraju, a vrijednosti izvan opsega izazivaju izuzetak. Metoda getEffectiveDamage u osnovnoj klasi Enemy vraća vrijednost damage. Klasa **MeleeEnemy** nasljeđuje Enemy i koristi osnovnu implementaciju napada. Klasa **BossEnemy** redefiniše getEffectiveDamage tako da vraća dvostruku vrijednost. Metoda toString() u BossEnemy može naglasiti da se radi o boss neprijatelju.

Glavna klasa Game sadrži jedan objekat klase Player, listu neprijatelja tipa ArrayList<Enemy> i log događaja tipa ArrayList<String>. Potrebno je implementirati metode: checkCollision(Player p, Enemy e), koja koristi intersects; decreaseHealth(Player p, Enemy e), koja smanjuje health igrača, ali ne ispod nule, i bilježi događaj u log; addEnemy(Enemy e), koja dodaje neprijatelja i bilježi događaj; findByType(String query), koja pronalazi sve neprijatelje čiji tip sadrži traženi tekst, ne razlikujući velika i mala slova; collidingWithPlayer(), koja vraća listu neprijatelja u koliziji sa igračem; i resolveCollisions(), koja prolazi kroz sve neprijatelje i primjenjuje logiku kolizije i smanjenja zdravlja, bilježeći svaku promjenu u log.

U okviru klase Game dodati i statičku metodu loadEnemiesFromCSV(String filePath), koja učitava CSV fajl u kome se nalaze podaci o neprijateljima. Fajl je strukture gdje se svaki neprijatelj

opisuje tipom, klasom ("melee" ili "boss"), pozicijom, oblikom kolajdera i atributima napada. Na osnovu pročitane informacije kreira se odgovarajući objekat tipa MeleeEnemy ili BossEnemy, uz upotrebu odgovarajućeg kolajdera (RectangleCollider ili CircleCollider). Ako se naiđe na neispravan zapis, baca se IllegalArgumentException. Dat je primjer CSV fajla na edukuj.me (Nedelja 12, enemies.csv). **Napomena**: zbog lakšeg parsiranja fajla dat je CSV. Za ispit je vjerovatnije da će to biti JSON fajl ili txt ali vjerovatno ne sa ovako složenom strukturom.

Nakon kreiranja osnovne logike igre, kreirati grafički korisnički interfejs (GUI) u Javi, preko Swing, koji omogućava unos podataka za igrača. GUI treba da sadrži tekstualna polja za unos imena, početnog zdravlja i pozicije (x i y), kao i izbor između pravougaonog i kružnog kolajdera. Moguće je koristiti fiksne vrijednosti dimenzija, na primjer pravougaonik 32x32 ili krug sa poluprečnikom 16. Nakon unosa, klikom na dugme za potvrdu Pokreni igru, GUI treba da kreira objekat Player na osnovu unesenih podataka i pozove metodu za učitavanje neprijatelja iz CSV fajla. Zatim treba da se pokrene logika kolizije i na ekranu prikažu rezultati: trenutni status igrača, lista neprijatelja, lista onih koji su u koliziji sa igračem, kao i kompletan log događaja. Rezultati se mogu prikazati u komponenti tipa JTextArea. U glavnom dijelu programa potrebno je da aplikacija pokreće GUI, čime se korisniku omogućava unos podataka o igraču, dok se ostatak igre automatski generiše iz CSV fajla. GUI sloj ne treba da zna kako funkcionišu kolizije i logika igre, te odgovornosti su delegirane klasama modela i klasi Game. Ako je Player poražen ili pobijedio, treba da iskoči odgovaraću MessageDialog.