SEW-Inhalte 3. Klasse

Objektorientierte Entwicklung

Das gesamte Projekt ist objektorientiert strukturiert. Es gibt Klassen für Gegner (z. B. Enemy, FlowEnemy, MetalBloon), Türme (Tower, SniperTower, MagicTower), Projektile (Projectile, MagicProjectile) und Spielkomponenten (Game, Path).

Beispiel:

public class Enemy {

protected double x, y;

protected int health;

public void takeDamage(int amount) {

health -= amount;

}

}

Rekursion

Die Pfadfindung und das Platzieren von Türmen mit automatischer Nachbesserung bei ungültiger Position verwenden rekursive Hilfsfunktionen.

```
private void tryPlaceTower(MouseEvent e) {
  Point2D click = new Point2D(e.getX(), e.getY());
  if (paused) {
    showMessage("Spiel ist pausiert – keine Platzierung möglich.");
    return;
  }
```

Collections

Sämtliche Gegner, Türme und Projektile werden in Collections verwaltet (z. B. ArrayList, HashMap).

List<Enemy> enemies = new ArrayList<>();

Map<Tower, TowerType> towerTypeMap = new HashMap<>();

Exceptions (Präzise Fehlermeldungen)

Fehlerhafte Platzierungen oder ungültige Spielaktionen erzeugen gezielte Ausgaben.

if (!canPlaceTowerHere(x, y)) {

throw new IllegalArgumentException("Turm darf nicht im Schussbereich eines anderen Turms platziert werden.");

}

Streams

Java-Streams wurden verwendet, um z. B. die Gegnerliste zu filtern.

```
enemies.stream()
.filter(e -> e.isVisible() && !e.isDead())
.forEach(e -> e.update());
```

Threads

Die Spiellogik (Zeitsteuerung der Wellen) basiert auf einem JavaFX-AnimationTimer, der intern mit Threads arbeitet. Manche Gegner wie FlowEnemy arbeiten zusätzlich mit einem Cooldown-Thread für Spezialbewegung.

2. Sonstiges

Style-Guides

Das Projekt folgt dem Google Java Style Guide. Klassen, Methoden und Variablen sind konsistent benannt.

```
public class SniperTower extends Tower {
private static final int RANGE = 300;
}
```

Einfachheit

Die Architektur ist bewusst einfach gehalten. Ein GameLoop ruft zyklisch updateGame() auf. Zusätzlich gibt es klar getrennte Klassen für GUI, Spiellogik und Gegner.

Laufzeiteffizienz

Effizienz wurde durch frühe Abbrüche in Schleifen, Nutzung von HashMap für schnelle Zugriffe und removeIf optimiert.

```
projectiles.removeIf(p -> p.hasExpired());
```

Zusatzfeatures

Unsichtbare Gegner (InvisibleEnemy) und nur von bestimmten Türmen sichtbar

Upgrade-Menü für Bogenschützenturm mit Auswahl von Damage oder Range

MetalBloon: Besondere Hülle, nur durch Feuer/Bombe zerstörbar

RegenEnemy: Gegner mit Regeneration

Restart-Button mit Reset der Spielwelt

FlowEnemy: Gegner, der fliegt und nur von bestimmten Türmen erkannt wird