Rapport formell granskning av INTE Rouglike

Benjamin Avery
Ludvig Svee
Sebastian Gieser
Sebastian Järvitalo

Character.java

Medlemsvariablerna turnsystem och inventory kan vara final.
tostring saknar @Override annotering.
Ta bort konstruktor för bakåtkompatibilitet:

```
public Player(int speed, int health, int damage, int x, int y, TurnSystem ts)
public Player(int speed, int health, int damage, TurnSystem ts){
```

Enemy.java

takedamage saknar @Override annotering.
Har duplicerad kod som kan flyttas till basklassen.

EnemyAI.java

random kan vara final.

Inventory.java

Datastrukturerna: percentage, items, itemsbystat kan vara final.
Annotering @override saknas på tostring.

IO.java

Ändra abstrakta klassen IO till ett interface - saknar implementation och därav representerar den mer ett kontrakt än en abstrakt klass.

Item.java

Medlemsvariablerna: name, effect och value kan vara final.
Enums bör vara högst upp.

toString saknar annotering @Override.
tostring: negativ procent saknar tecken samt att det borde vara value istället - beräknar egentligen procentenheter.

MapController.java

För långa rader
Medlemsvariabeln map kan vara final.

Mappable.java

Det är ingen bra design att Mappable är tom för att enda användningen nu är för starkare typtest än att spara dem som Object i datastrukturer.

MappableTypeWrapper.java

Alla medlemsvariabler utom currentquantity kan vara final.

Position.java

CardinalDirection ska vara versaler för den representerar en konstant.
CardinalDirectionPermission ska vara versaler för den representerar en konstant.
Position borde vara immutable.

Room.java

För stor klass, tyder på behov av refaktorisering (bryter mot Single-responsibility Principle).
Konstruktor är för legacykod och borde tas bort: public Room(Position worldPosition, RoomSpace roomSpace).
Majoriteten av metoderna kan brytas ut till hjälpmetoder i en separat klass.

RoomCreator.java

För långt metodnamn: getCardinalDirectionPermissionChoice.

RoomCreatorBuilder.java

Byt namn på addEnemy till addEnemyTemplate för att visa på att det är mallar som skapas.
Byt namn på addEnemy till addItemTemplate för att visa på att det är mallar som skapas.

testType är helt generisk och borde refaktoriseras ut från klassen.

RoomSpace.java

Kod måste ta hänsyn till hur array index implementeras i RoomSpace (roomspace:4 = 0, 1, 2, 3 men get width/height returnerar mängden). Detta bryter OO-principen om *Dependency Inversion principle*.

TUI.java

requestMove har en alltför hög cyklomatisk komplexitet och bör refaktoriseras till en FSM.
Inmatade värden från getInput FromUser borde konverteras till gemener.

TurnSystem.java

Medlemsvariabeln io kan vara final.
Ta bort metoden move för den används inte längre.
StartTurn har magic numbers, dvs konstanta icke namngivna värden.
Metodnamnet: doorIsInMappable är missvisande och metoden behöver också delas upp i flera enskilda metoder.
Fånga IllegalArgumentException i stället för Exception. Risken är annars att fel maskeras samt att en lägre nivå hanterar fel som borde delegeras uppåt i hierarkin.
Namnbyte på getnewPosition till getNewPosition.
Refaktorisera startTurn till en FSM för att minska den cyklomatiska komplexiteten.

WallWalkerAI.java

Lägga till get/set lastDir i enemyAI för att flytta upp lastDir från wallWalker.