Metod	Fel	Prio
Character.java	Medlemsvariablerna turnsystem och inventory kan vara final.	0
Character.java	tostring saknar @Override annotering.	0
Character.java	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)	1
Enemy.java	takedamage saknar @Override annotering.	0
Enemy.java	Har duplicerad kod som kan flyttas till basklassen.	1
EnemyAI.java	random kan vara final.	0
Inventory.java	Datastrukturerna: percentage, items, itemsbystat kan vara final.	0
Inventory.java	Annotering @override saknas på tostring.	0
IO.java	Ändra abstrakta klassen IO till ett interface - saknar implementation och därav representerar den mer ett kontrakt än en abstrakt klass.	1
Item.java	Medlemsvariablerna: name, effect och value kan vara final.	0
Item.java	Enums bör vara 2st upp.	0
Item.java	toString saknar annotering @Override.	0
Item.java	tostring: negativ procent saknar tecken samt att det borde vara value istället - beräknar egentligen procentenheter.	1
MapController.java	För långa rader	0
MapController	Medlemsvariabeln map kan vara final.	0
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.	0
MappableTypeWrapper.java	Alla medlemsvariabler utom currentQuantity kan vara final.	0
Position.java	CardinalDirection ska vara versaler för den representerar en konstant.	0
Position.java	CardinalDirectionPermission ska vara versaler för den representerar en konstant.	0
Position.java	Position borde vara immutable.	2
Room.java	För stor klass, tyder på behov av refaktorisering (bryter mot Single-responsibility Principle).	1
Room.java	Konstruktor är för legacykod och borde tas bort: public Room(Position worldPosition, RoomSpace roomSpace).	2

Metod	Fel	Prio
Room.java	Majoriteten av metoderna kan brytas ut till hjälpmetoder i en separat klass.	1
RoomCreator.java	För långt metodnamn: getCardinalDirectionPermissionChoice.	0
RoomCreatorBuilder.java	Byt namn på addEnemy till addEnemyTemplate för att visa på att det är mallar som skapas.	1
RoomCreatorBuilder.java	Byt namn på addEnemy till addItemTemplate för att visa på att det är mallar som skapas.	1
RoomCreatorBuilder.java	testType är helt generisk och borde refaktoriseras ut från klassen.	0
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*.	2
TUI.java	requestMove har en alltför 2 cyklomatisk komplexitet och bör refaktoriseras till en FSM.	2
TUI.java	Inmatade värden från getInput FromUser borde konverteras till gemener.	1
TurnSystem.java	Medlemsvariabeln io kan vara final.	0
TurnSystem.java	Ta bort metoden move för den används inte längre.	1
TurnSystem.java	StartTurn har magic numbers, dvs konstanta icke namngivna värden.	2
TurnSystem.java	Metodnamnet: doorIsInMappable är missvisande och metoden behöver också delas upp i flera enskilda metoder.	2
TurnSystem.java	Fånga IllegalArgumentException i stället för Exception. Risken är annars att fel maskeras samt att en 0 re nivå hanterar fel som borde delegeras uppåt i hierarkin.	2
TurnSystem.java	Namnbyte på getnewPosition till getNewPosition.	0
TurnSystem.java	Refaktorisera startTurn till en FSM för att minska den cyklomatiska komplexiteten.	2
WallWalkerAI.java	0 ga till get/set lastDir i enemyAI för att flytta upp lastDir från wallWalker.	1