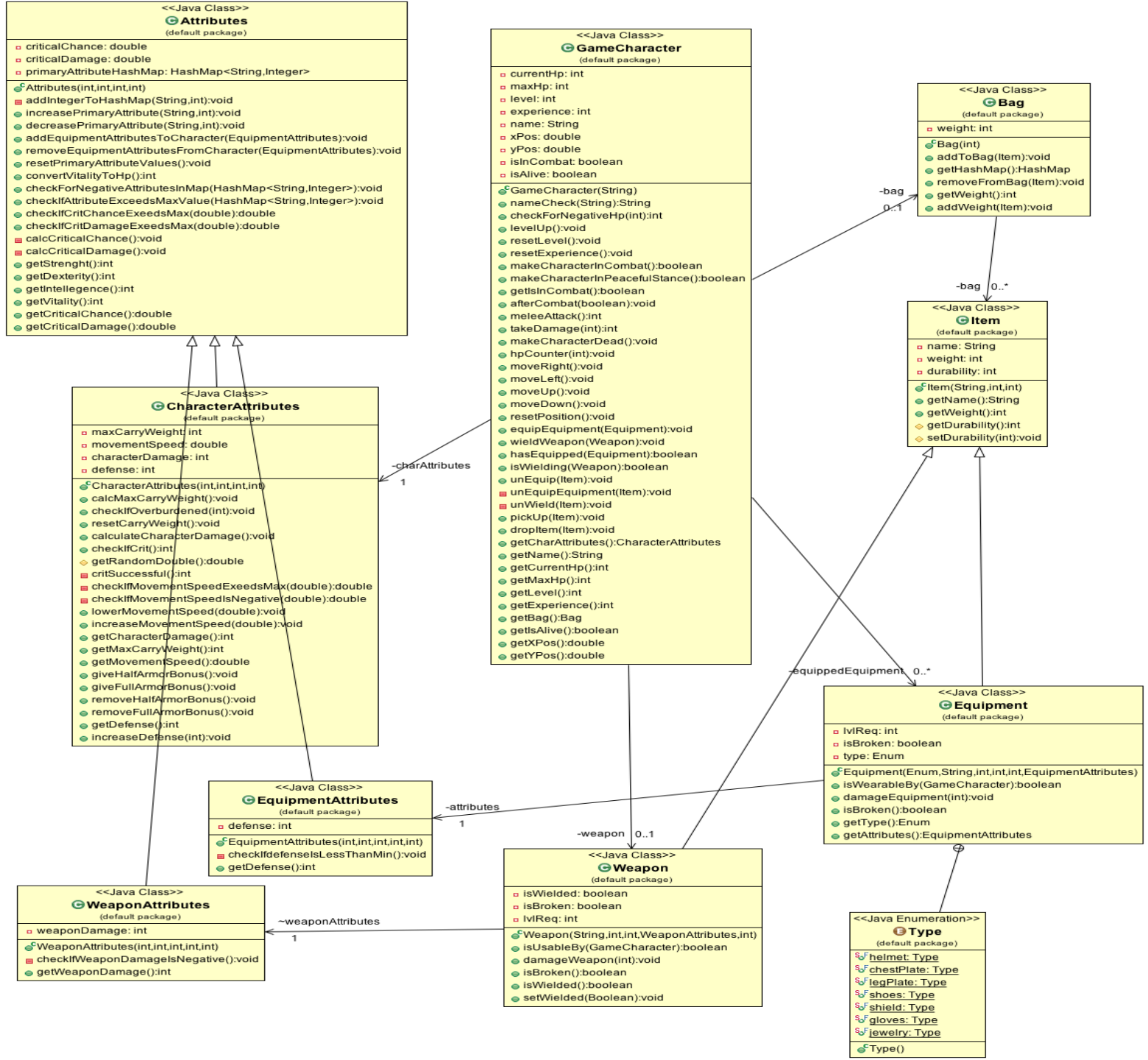
Grupp nr: 1

- Lukas Varli Lukas.varli@hotmail.com
- Peter Yakob peteryakobte11@gmail.com
- Emil Rosell emil.rosell@outlook.com
- Oscar Törnquist tornquist93@gmail.com
- Robert Grubesic robertgrubesic@hotmail.com

Verktyg

Versionshanteringssystem: GitHub
Byggscript: Ant
Enhetsramverk: Junit 4
Kodkritiksystem: FindBugs
Täckningsgrad: IntelliJ IDEA code coverage runner
Statiska mått: Metrics Reloaded
Profiler: NetBeans Profiler

Slutlig design



TDD-exempel: Lukas

Testkod

```
@Test
public void testCheckExperienceAfterCharacterIsDead() {
    GameCharacter mainCharacter = new GameCharacter("kalle");
    mainCharacter.makeCharacterInCombat();
    int currentHp = mainCharacter.getCurrentHp();
    mainCharacter.hpCounter(currentHp);
    mainCharacter.makeCharacterDead();
    assertEquals(0, mainCharacter.getExperience());
}
@Test
public void testResetPositionX() {
    GameCharacter mainCharacter = new GameCharacter("kalle");
    mainCharacter.moveRight();
    mainCharacter.makeCharacterInCombat();
    mainCharacter.hpCounter(mainCharacter.getCurrentHp());
    mainCharacter.makeCharacterDead();
    assertEquals(20.0, mainCharacter.getXPos(), 0.1);

}

@Test
public void testResetPositionY() {
    GameCharacter mainCharacter = new GameCharacter("kalle");
    mainCharacter.moveRight();
    mainCharacter.moveDown();
    mainCharacter.makeCharacterInCombat();
    mainCharacter.hpCounter(mainCharacter.getCurrentHp());
    mainCharacter.makeCharacterDead();
    assertEquals(10.0, mainCharacter.getYPos(), 0.1);
```

Koden som testas

```
public void makeCharacterDead() {
    if (isInCombat && currentHp == 0) {
        isAlive = false;
        resetExperience();
        resetPosition();
    }
}
```

TDD-exempel: Lukas

Testkod

```
@Test
public void testIsInCombatAfterFight() throws Exception {
    GameCharacter character = new GameCharacter("Gubbe");
    character.afterCombat(true);
    assertEquals(false, character.getIsInCombat());
}

@Test
public void testExperienceAfterFight() throws Exception {
    GameCharacter character = new GameCharacter("Gubbe");
    character.afterCombat(true);
    assertEquals(10, character.getExperience());
}

@Test
public void testLevelAfterFight() throws Exception {
    GameCharacter character = new GameCharacter("Gubbe");
    character.afterCombat(true);
    assertEquals(1, character.getLevel());
}
```

Koden som testas

```
public void afterCombat(boolean isInCombat) {
    if (isInCombat) {
        makeCharacterInPeacefulStance();
        experience += 10;
        if (experience > 30) {
            levelUp();
            resetExperience();
        }
    }
}
```

TDD-exempel: Emil

Testkod

```java
@Test(expected = IllegalArgumentException.class)
public void testNullMapName() {
    Map map = new Map( mapName: null,  mapHeight: 500,  mapWidth: 500);
    assertEquals( expected: null, map.getMapName());
}

@Test(expected = IllegalArgumentException.class)
public void testEmptyString() {
    Map map = new Map( mapName: "",  mapHeight: 500,  mapWidth: 500);
    assertEquals( expected: "The forbidden forest", map.getMapName());
}

@Test
public void testMapHeight() {
    Map map = new Map( mapName: "The forbidden forest",  mapHeight: 500,  mapWidth: 500);
    assertEquals( expected: 500, map.getMapHeight());

}

@Test(expected = IllegalArgumentException.class)
public void testNegativeHight() {
    Map map = new Map( mapName: "The forbidden forest",  mapHeight: -10,  mapWidth: 500);
    assertEquals( expected: 500, map.getMapHeight());

}

@Test
public void testMaxHeigth() {
    Map map = new Map( mapName: "The forbidden forest",  mapHeight: 1000,  mapWidth: 500);
    assertEquals( expected: 1000, map.getMapHeight());
}
```

Koden som testas

```java
public Map(String mapName, int mapHeight, int mapWidth) {
    this.mapName = mapName;
    if (mapName == null || mapHeight < 0 || mapWidth < 0 || mapName.isEmpty()) {
        throw new IllegalArgumentException("Something went wrong");
    } else if (mapWidth > mapMaxWidth || mapHeight > mapMaxHeight) {
        throw new IllegalArgumentException("Something went wrong");
    } else {
        this.mapWidth = mapWidth;
        this.mapHeight = mapHeight;
    }

}
```

TDD-exempel: Emil

Testkod

```
@Test
public void testWeight() {
    Bag bag = new Bag( weight: 10);
    EquipmentAttributes attributes = new EquipmentAttributes( defense: 5,  strenght: 5,  dexterity: 5,  intellegence: 5,  vitality: 5);
    Equipment ring = new Equipment(Equipment.Type.jewelry,  name: "Ring",  lvlReq: 2,  durability: 10,  weight: 10, attributes);
    bag.addToBag(ring);
    assertEquals( expected: 20, bag.getWeight());
}
```

Koden som testas

```
public void addWeight(Item item) {
    weight += item.getWeight();
}
```

TDD-exempel: Robert

Testkod

```
// TESTS EQUIPMENT
@Test
public void isWearableByTest() {
    GameCharacter player = new GameCharacter( name: "Isaac"); // New player created level 1
    EquipmentAttributes attributes = new EquipmentAttributes( defense: 5,  strenght: 5,  dexterity: 5,  intellegence: 5,  vitality: 5);
    Equipment ring = new Equipment(Equipment.Type.jewelry,  name: "Ring",  lvlReq: 2,  durability: 10,  weight: 10, attributes);

    player.levelUp( amount: 1); // levels upp player to lvl 2

    assertEquals( expected: true, ring.isWearableBy(player)); // tests if player can wear the ring
}

@Test
public void isNotWearableByTest() {
    GameCharacter player = new GameCharacter( name: "Isaac"); // New player created level 1
    EquipmentAttributes attributes = new EquipmentAttributes( defense: 5,  strenght: 5,  dexterity: 5,  intellegence: 5,  vitality: 5);
    Equipment shoes = new Equipment(Equipment.Type.shoes,  name: "fast shoes",  lvlReq: 2,  durability: 10,  weight: 10, attribut

    assertEquals( expected: false, shoes.isWearableBy(player)); // tests if player can wear the ring
}

@Test
public void brokenWearableTest() {
    GameCharacter player = new GameCharacter( name: "Isaac"); // New player created level 1
    EquipmentAttributes attributes = new EquipmentAttributes( defense: 5,  strenght: 5,  dexterity: 5,  intellegence: 5,  vitality: 5);
    Equipment ring = new Equipment(Equipment.Type.jewelry,  name: "Ring",  lvlReq: 2,  durability: 10,  weight: 10, attributes);
    ring.damageEquipment(10);
    assertEquals( expected: false, ring.isWearableBy(player)); // tests if player can wear the ring
```

Koden som testas

```
public boolean isWearableBy(GameCharacter player) {
    if (isBroken)
        return false;
    if (player.getLevel() >= lvlReq)
        return true;
    else
        return false;
}
```

TDD-exempel: Robert

Testkod

```
+    @Test
+    public void denyPickUpIfInCombatTest(){
+        GameCharacter player = new GameCharacter("Oscar");
+        WeaponAttributes weaponAttributes = new WeaponAttributes(10, 10, 10, 10, 10);
+        Weapon swordOfDoom = new Weapon("Sword of doom", 10, 10, weaponAttributes, 0);
+        player.enterCombat();
+        player.pickUpItem(swordOfDoom);
+
+        Bag bag = player.getBag();
+
+        assertFalse(bag.getHashMap().containsKey(swordOfDoom.getName()));
+
+
+    }
```

Koden som testas

```
     public void pickUpItem(Item item) {
+        if(isInCombat){
+            System.out.println("Cant pick up while in combat");
+
+        }else{
+            bag.addToBag(item);
+            charAttributes.checkIfOverburdened(bag.getWeight());
+
+        }
+
     }
```

TDD-exempel: Oscar

```java
@Test
public void testCharacterYPosMoreThanWidth() {
    Map theWoods = new Map( mapName: "theWoods",  mapHeight: 1000,  mapWidth: 1000);
    GameCharacter g = new GameCharacter( name: "g");

    g.getCharAttributes().increaseMovementSpeed( amount: 1.0);
    for (int i = 1; i < 1000; i++) {
        g.moveUp();
    }

    assertFalse(theWoods.isWithinMap(g));
}

@Test
public void testCharWithinMap() {
    Map theWoods = new Map( mapName: "theWoods",  mapHeight: 1000,  mapWidth: 1000);
    GameCharacter g = new GameCharacter( name: "g");
    for (int i = 1; i < 500; i++) {
        g.moveUp();
        g.moveRight();
    }
    assertTrue(theWoods.isWithinMap(g));
}

@Test
public void testNotWithinMap() {
    Map theWoods = new Map( mapName: "theWoods",  mapHeight: 1000,  mapWidth: 1000);
    GameCharacter g = new GameCharacter( name: "g");
    for (int i = 1; i < 12000; i++) {
        g.moveUp();
        g.moveRight();
    }
    assertFalse(theWoods.isWithinMap(g));
}

@Test
public void testResetWithinMap() {
    Map theWoods = new Map( mapName: "theWoods",  mapHeight: 1000,  mapWidth: 1000);
    GameCharacter g = new GameCharacter( name: "g");
    for (int i = 1; i < 12000; i++) {
        g.moveUp();
        g.moveRight();
    }
    g.makeCharacterInCombat();
    g.hpCounter(g.getCurrentHp());
    g.makeCharacterDead();
    assertTrue(theWoods.isWithinMap(g));
}
```

Koden som testas

```java
public boolean isWithinMap(GameCharacter character) {
    double xPos = character.getXPos();
    double yPos = character.getYPos();
    if (xPos < 0 || yPos < 0) {
        return false;
    } else if (xPos > mapWidth || yPos > mapHeight) {
        return false;
    }
    return true;

}
```

TDD-exempel: Oscar

Testkod

```
225 +   @Test
226 +   public void moveRightWithHigherSpeed() {
227 +       GameCharacter g = new GameCharacter("Oscar");
228 +       g.getCharAttributes().increaseMovementSpeed(0.5);
229 +       g.moveRight();
230 +       g.moveRight();
231 +       assertEquals(23, g.getXPos(), 0.1);
232 +   }
233 +
234 +   @Test
235 +   public void moveRightAndChangeSpeedInTheMiddle() {
236 +       GameCharacter g = new GameCharacter("Oscar");
237 +       g.moveRight();
238 +       g.getCharAttributes().increaseMovementSpeed(0.5);
239 +       g.moveRight();
240 +       g.getCharAttributes().increaseMovementSpeed(0.5);
241 +       g.moveRight();
242 +       assertEquals(24.5, g.getXPos(), 0.1);
243     }
```

Koden som testas

```
124 +   public void moveRight() {
125 +       xPos += charAttributes.getMovementSpeed();
126 +   }
```

TDD-exempel: Peter

Testkod

```
// 101 Dexterity results in critChance exceeding the max limit which is 1.0 (100%)
@Test(expected = AssertionError.class)
public void ExeedsMaxCritChance() throws Exception{
    CharacterAttributes c = new CharacterAttributes(10,101,10,10,10);
    assertEquals(1.001, c.getCriticalChance(), 0.0001);
}
```

Koden som testas

```
public double checkIfCritChanceExeedsMax(double critChance){
    if(critChance > 1)
        return 1;
    else
        return critChance;
}
 private void calcCriticalChance() {
     criticalChance = dexterity / 100;
     criticalChance = checkIfCritChanceExeedsMax(criticalChance);
 }

public double getCriticalChance(){
    calcCriticalChance();
    return criticalChance;
}
```

TDD-exempel: Peter

Testkod

```
6  +    @Test
7  +    public void increasePrimaryAttributeTest() throws Exception {
8  +        CharacterAttributes c = new CharacterAttributes(10, 10, 10, 10);
9  +        c.increasePrimaryAttribute("Strength", 7);
10 +        assertEquals(17, c.getStrenght(), 0);
11 +    }
12 +
```

Koden som testas

```
26 +    public void increasePrimaryAttribute(String s, int value) {
27 +        if (primaryAttributeHashMap.containsKey(s)) {
28 +            primaryAttributeHashMap.put(s, primaryAttributeHashMap.get(s) + value);
29 +        }
30 +    }
```

TDD erfarenheter

En av de bästa erfarenheten av TDD är att man får tänka till ordentligt när man skriver sin kod, det ger dig vissa begränsningar när man väl ska börja koda vilket hjälper en att hålla den röda tråden och ser till att du kan strukturera din kod korrekt. Men ibland kan den första tanken bli fel vilket gör att man får tänka om på hur man skrivit sin testkod/testfall och detta leder till att man kanske skriver enklare eller bättre kod än som var avsiktligt.

Testfallsdesign ekvivalensklasser

- Vi har valt att tillämpa ekvivalensklassuppdelning på klassen Map. När man skapar en karta anges namnet på vad kartan ska heta samt bredden och höjden. Man kan tydligt dela upp detta i grupper och skapa begränsningar, därför har vi valt att utföra en ekvalensklassuppdelning här. Vi har delat upp ekvivalensklasserna i mapName, mapHeight och mapWidth. Där vi har gjort tydliga begränsningar så att man inte ska kunna skapa en för stor karta samt en karta som inte finns. Vi har även gjort att man inte ska kunna skapa en karta utan namn.

Ekvivalensklassuppdelning för klassen Map

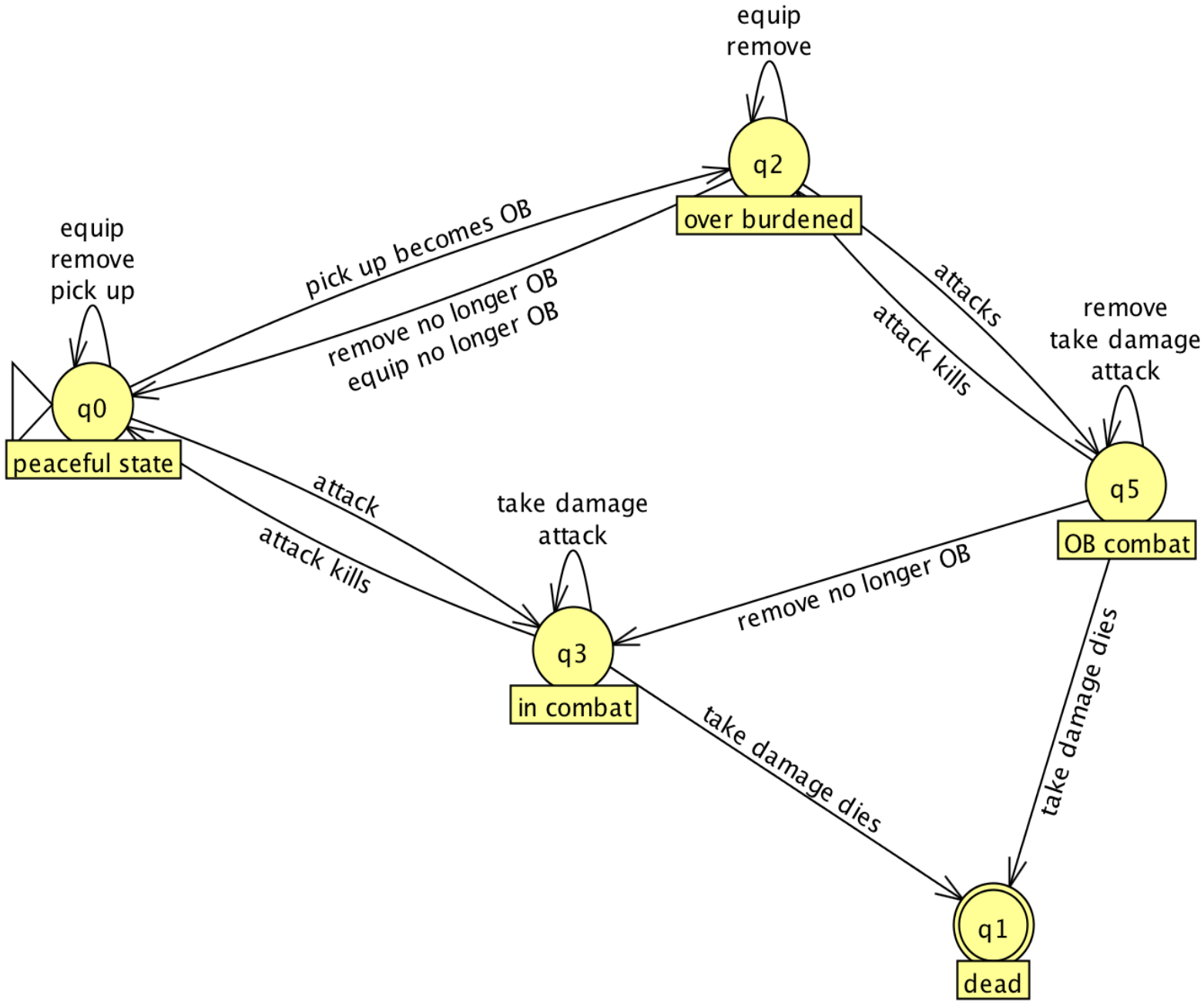
mapName	mapHeight	mapWidth
~~**N1**~~ null	~~**H1**~~ <0	~~**W1**~~ <0
~~**N2**~~ ""	~~**H2**~~ 0-max	~~**W2**~~ 0-max
~~**N3**~~ Kort sträng	~~**H3**~~ >max	~~**W3**~~ >max
~~**N4**~~ Lång sträng		

ID	In	Ut	Täckta
1	"Mordor", 500, 500	OK	N3, H2, W2
2	"The forbidden forest", 500, 500	OK	N4, H2, W2
3	"", 500, 500	IAE	N2
4	null, 500, 500	NPE	N1
5	"Mordor", -10, 500	IAE	H1
6	"Mordor", 1500, 500	IAE	H3
7	"Mordor", 500, -10	IAE	W1
8	"Mordor", 500, 1500	IAE	W3

Tillståndsmaskiner

- Vi har valt att tillämpa tillståndsmaskinen på karaktären då den är i peaceful state, overburdened, in combat, in combat overburdned och dead. Vi har valt att nå branch coverage dvs täcka alla tillstånd samt vägar. Vi tyckte det var intressant hur karaktären kan plocka upp items, bli overburdened och ändå ha möjligheten att gå in i combat. Sedan var det nödvändigt att testa om karaktären kan ta bort item och gå in i vanlig combat eller gå tillbaka till peaceful state. Vi såg även till att karaktären inte ska kunna plocka upp items då den är i combat.

Tillståndsmaskinen



Testfall

ID	Beskrivning	Täckta tillstånd	Täckta övergångar
1	1. Plockar upp item 2. Kastar bort item ur väskan 3. Plockar upp item 4. Tar på sig item 5. Plockar upp item och blir OB 6. Tar bort och går tillbaka i peaceful state 7. Plockar upp item och blir OB 8. Tar på sig item och går tillbaka i peaceful state 9. Plockar upp item och blir OB	Peaceful state, Overburdened, OB combat, In combat Dead	Equp, Remove, Pick up, Pick up becomes OB, Remove no longer OB, Equip no longer OB Attacks, Attack kills, Take damage, Remove no longer OB, Take damage dies

Testfall

ID	Beskrivning	Täckta tillstånd	Täckta övergångar
1	10. Attackerar och går in i OB combat 11. Attack dödar och går tillbaka till OB 12. Attackerar och går in i OB combat 13. Tar damage 14. Attackerar 15. Tar bort item 16. Tar bort item och går in i vanlig combat 17. Tar damage och dör.	Peaceful state, Overburdened, OB combat, In combat Dead	Equp, Remove, Pick up, Pick up becomes OB, Remove no longer OB, Equip no longer OB Attacks, Attack kills, Take damage, Remove no longer OB, Take damage dies

Testfall

ID	Beskrivning	Täckta tillstånd	Täckta övergångar
2	1. Attackerar och går in i combat 2. Attackerar 3. Tar damage 4. Attack dödar och går in i peaceful state 5. Tar på sig item och blir OB 6. Attackerar och går in i OB combat 7. Tar damage och dör	Peaceful state, In combat, Over burdened, OB combat, dead	Attack, Take damage, Attack kills, Pick up becomes OB, Attacks, Take damage dies

Granskning

Vi har utfört en formell teknisk inspektion där var och en av gruppmedlemmarna systematisk gått igenom klassen ”GameCharacter” som är primärt skriven av Lukas Varli(Författare). Vi har använt oss av checklistan(Från Seminaretillfälle Två) som vår röda tråd för att underlätta granskningen och se till att vi jobbar inom tydliga ramar.

Den primära anledningen för att vi valde just ”GameCharacter” var för att den klassen mätt i antalet metoder är den största klassen och att den kan ses som en integrationshub för hela programmet, vilket ökar risken till att den har fler defekter än de andra klasserna.

Granskningsrapport

En metod ”hpCounter” hittades som inte uppfyllde någon funktion. Den användes tidigare i projeket för testing av HP men ersattes senare av takeDamage som uppfyller samma funktion på ett effektivare sätt.

Nament på metoden unWield bör specificeras, till exempel unWieldWeapon för att undvika förvirring.

Namnet på metoden nameCheck bör specificeras. För att tydliggöra dess funktion kan man ändra till checkValidName

Namnet på metoden makeCharachterInPeacefulStance skulle kunna förenklas till notInCombat eller outOfCombat.

En metod ”hpCounter” hittades som inte uppfyllde någon funktion. Den användes tidigare i projeket för testing av HP men ersattes senare av takeDamage som uppfyller samma funktion på ett effektivare sätt.

Metoden hasEquipped() kan förenklas genom att slå ihop if sattserna med en &&

Metod namnet “makeCharacterInCombat” & “MakeCharacterInPeacefulStance” är onödigt komplicerade och skulle kunna döpas om till t.ex. “enterCombat” & “exitCombat”.

Metod namnet “getIsInCombat” skulle kunna döpas om till “isCharacterInCombat”.

Metod namnet “makeCharacterDead” skulle kunna döpas om till “killCharacter” eller “characterDies”.

Metod namnet “equipEquipment” skulle kunna döpas om till “useEquipment”.

Metod namnet “isWielding” skulle kunna döpas till “isCharacterWieldingWeapon” för att bättre beskriva dess funktion.

Metod namnet “hasEquipped” skulle kunna döpas om till “isEquipmentInUse”.

Metod namnet “unWield” skulle kunna döpas om till “unWieldWeapon” för att specificera dess funktion.

Metod namnet “pickUp” skulle kunna döpas till “pickUpItem” för att specificera dess funktion.

Metoden “resetLevel” kan göras private

Metoden “resetExperience” kan göras private

Granskningsrapport

- De felen som vi har hittat anser vi inte vara så allvarliga eftersom de inte påverkar funktionen av programmet. De flesta felen handlar om dåliga namnval på metoderna samt nån metod som inte uppfyller någon funktion alls. Två fel som är lite allvarligare än de andra är metoderna som kan göras private istället för public då de ändast används i klassen själv.

Erfarenheter av granskning

Den största erfarenheten vi kunde dra av att utföra en formell teknisk inspektion var hur effektiv den var när det kommer till att hitta fel. Att först enskilt gå igenom koden noggrant steg för steg och sedan gå igenom den tillsammans visade sig vara väldigt effektivt. Det gav oss en möjlighet att gå igenom koden för att se till att allting ser bra ut och fungerar som det ska. Det är ett bra sätt att se hur varandras kod ser ut i minsta detalj och hur man har jobbat.

Kodkritiksystem

- Några av felen vi hittade i FindBugs visar att under granskningen så fokuserade vi på annat än vad som kan va relevant för funktionen i programmet. Huvudfokus i granskningen var att göra koden lätt att förstå och refaktorera metoder med svåra namn. FindBugs hittade mer fel såsom ”\n” kunde ersättas med ”%n” i en printf() för att säkra funktionaliteten över flera operativsystem och miljöer.
- Kodkritiksystemet har varit inställt på högsta känslighet både före och efter granskningen, dock så hittade det inte något kritiskt som behövdes åtgärdas när vi korrigerat allt från granskningrapporten samt första körningen av FindBugs.
- Det felet som FindBugs hittat är att returvärdet i getMaxHp() ignoreras, om vi tolkat uppgifterna rätt. I nuläget används inte metoden för mer än testning av variabeln maxHp, dock så kommer den vara till stor nytta när man i framtiden ska kunna lägga möjlighet att heala karaktären så livet inte överstiger max. Därav är det viktigt att ha det maximala värdet uppdaterat varje gång metoden kallas på.

Warning	Priority	Details
Return value of method without side effect is ignored	High	Return value of Attributes.convertVitalityToHp() ignored, but method has no side effect In file GameCharacter.java, line 269 In class GameCharacter In method GameCharacter.getMaxHp() Called method Attributes.convertVitalityToHp() At GameCharacter.java:[line 269] At GameCharacter.java:[line 269]

```
public int getMaxHp() {
    charAttributes.convertVitalityToHp();
    return maxHp;
}
```

Statiska mått

Vi har valt följande mått:

Coupling Factor (CF): 29,52%

Comment ratio (COM_RAT) : 3,69%

Lines of Product Code (LOCp) : 646

Lines of Test Code (LOCt): 950

Project metrics

project	CF	CLOC	COM_RAT	L(J)	L(XML)	LOC	LOCp	LOCt	TEST_RAT
project	29,52%	62	3,69%	1 596	39	1 678	646	950	56,62%

Statiska mått

Project metrics

project	CF	CLOC	COM_RAT	L(J)	L(XML)	LOC	LOCp	LOCt	TEST_RAT
project	29,52%	62	3,69%	1 596	39	1 678	646	950	56,62%

Coupling factor (CF)

Vi valde detta mått eftersom vi ville veta hur beroende klasserna är av varandra. I ett större projekt så hade detta mått haft en hög prioritet då en låg CF gör det enklare att bygga om systemet.

CF = Actual couplings / Maximum possible couplings

- Klass A är kopplad till klass B om A kallar på metoder eller får tillgång till variabler av B. I sin tur är B endast kopplad till A om B använder sig av metoder eller variabler från A. B är inte kopplad till A om det inte finns någon åtkomst från B till A.
- Max antal möjliga kopplingar händer när alla klasser är kopplade till/från alla andra klasser.
- Om inga klassser är kopplade, CF = 0%.
- Om alla klasser är kopplade till alla andra klasser, CF = 100%
- Koppling räknas i båda riktningarna. Om Klass A kallar på B och B kallar på A så räkas det som 2 kopplingar.

Höga värden på CF ger oftast en ökad defekt densitet, så med detta mått kan man ta reda på vilka delar av systemet som har större risk för att innehålla defekter. Höga värden av CF leder också till ökad komplexitet, begränsar förståelse av koden och gör det svårare att återanvända kod. Om man gör en snabb inspektion av vårt projektets klass-diagram så inser man att klassen GameCharacter är väldigt beroende av de andra klasserna vilket leder till en ökad coupling för projektet. Det kommer alltid finnas en mängd coupling i ett system, men det viktiga är att hålla den så låg som möjligt för att undvika framtida problem som kan uppstå.

I vårt projekt så är genomsnittet av kopplingar mellan klasserna 5,73 , den högsta är 13 (GameCharacter) och den lägsta är 2 (Map). Eftersom vi har ett relativt litet projekt så är 30% inte så högt värde, men hade vi fortsätt utveckla programmet så hade vi behövt hålla ett öga på kopplings faktorn så att den inte fortsätter att öka genom projektets utveckling.

Statiska mått

Project metrics

project	CF	CLOC	COM_RAT	L(J)	L(XML)	LOC	LOCp	LOCt	TEST_RAT
project	29,52%	62	3,69%	1 596	39	1 678	646	950	56,62%

Koppling mellan Objekt(CBO)

class	CBO
GameCharacter	13
GameCharacterTest	10
EquipmentAttributes	9
AllTests	8
ItemTest	8
Equipment	8
GameCharacterAttributesTest	8
Item	7
CharacterAttributes	6
Attributes	6
BagTest	6
Weapon	5
WeaponAttributes	5
Equipment.Type	5
Bag	4
Game	4
MapTest	4
AttributesTest	2
NotRandomCharacterAttributes	2
WeaponAttributesTest	2
EquipmentAttributesTest	2
Map	2
Total	
Average	5,73

Statiska mått

Project metrics

project	CF	CLOC	COM_RAT	L(J)	L(XML)	LOC	LOCp	LOCt	TEST_RAT
project	29,52%	62	3,69%	1 596	39	1 678	646	950	56,62%

Comment Ratio

Detta mått är ganska låg vilket kan indikera på bristfällig dokumentation i koden.

Mer dokumentation i koden gör det enklare att förstå syftet med klassen uppgifterna som den utför, vilket kan vara hjälpsamt om en annan aktör behöver lära sig hur koden fungerar.

Detta mått blir då viktigt när man vill spåra hur mycket av programmet man dokumenterat och om siffran blir för låg och man märker av det i de tidigaste stadierna av utvecklingsprocessen så kan man lätt se till att utvecklarna lägger mer tid på att dokumentera sin kod.

Ett högt värde kan även vara en indikation på många onödiga eller redundanta kommentarer eller att specifika bitar av koden är dåligt skriva och svåra att förstå.

Statiska mått

Project metrics

project	CF	CLOC	COM_RAT	L(J)	L(XML)	LOC	LOCp	LOCt	TEST_RAT
project	29,52%	62	3,69%	1 596	39	1 678	646	950	56,62%

Lines of codes

Dessa mått är inte så intressanta var för sig men tillsammans så tillåter de oss att se förhållandet mellan produkt koden samt test koden.

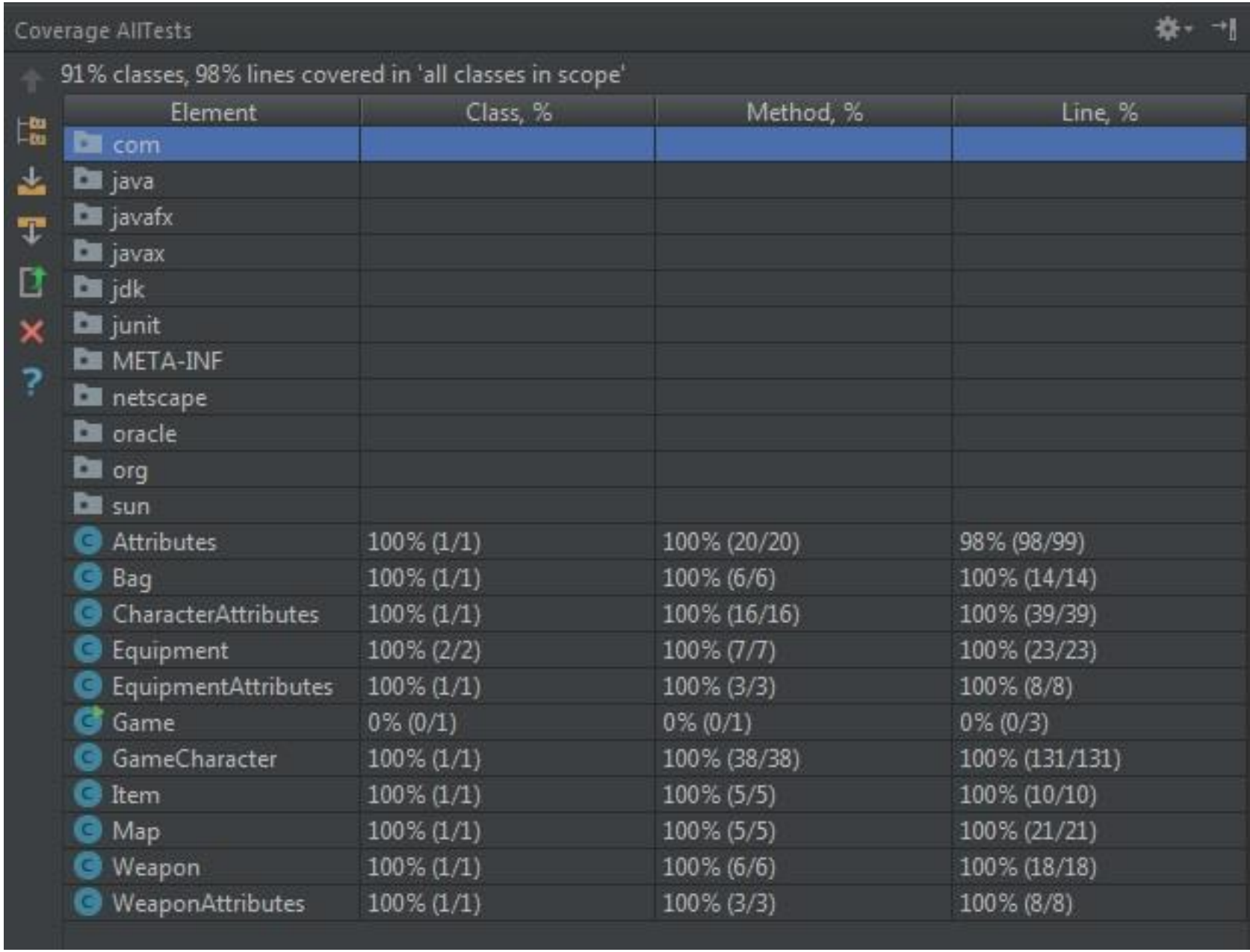
950/646 = 1.4705…

Förhållandet mellan de är ungefär 1:1.47

Vi har alltså ~47% mer test kod än produkt kod enligt diskussioner som vi har hittat online så är det normal att förhållandet brukar ligga mellan 1:1-1:3. Dessa mått är intressanta som en komplettering för coverage då code coverage bara visar om man täcker delar av koden, den berättar inte om man täckt alla möjliga vägar.

Ett lågt förhållande är en indikator av att man kanske bör lägga till fler testfall medans för höga värden kan vara en indikation på redundanta test eller gamla testfall som kanske bör tas bort eller refaktoreras.

Täckningsgrad

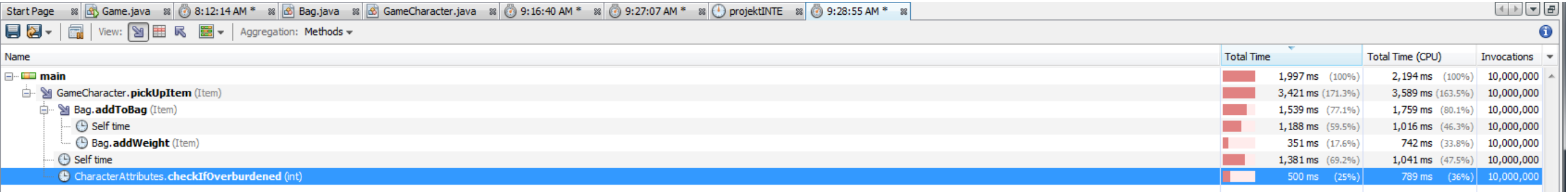


Element	Class, %	Method, %	Line, %
com			
java			
javafx			
javax			
jdk			
junit			
META-INF			
netscape			
oracle			
org			
sun			
Attributes	100% (1/1)	100% (20/20)	98% (98/99)
Bag	100% (1/1)	100% (6/6)	100% (14/14)
CharacterAttributes	100% (1/1)	100% (16/16)	100% (39/39)
Equipment	100% (2/2)	100% (7/7)	100% (23/23)
EquipmentAttributes	100% (1/1)	100% (3/3)	100% (8/8)
Game	0% (0/1)	0% (0/1)	0% (0/3)
GameCharacter	100% (1/1)	100% (38/38)	100% (131/131)
Item	100% (1/1)	100% (5/5)	100% (10/10)
Map	100% (1/1)	100% (5/5)	100% (21/21)
Weapon	100% (1/1)	100% (6/6)	100% (18/18)
WeaponAttributes	100% (1/1)	100% (3/3)	100% (8/8)

Profiler

I ett rougelike-spel så kan man ofta plocka upp många items samtidigt, därför tyckte vi att det skulle vara intressant att ta reda på hur metoden pickUpItems presterar när man försöker göra det.

Vi började med att i en for loop testa plocka upp 10 miljoner items



Resultatet blev ca 3.5 sekunder

3.5 sekunder ansåg vi skulle vara för lång tid att vänta för en spelare och därför skulle inte plocka up funktionen upplevas som att den utförs i ”real time”

Vi fortsatte köra profilern, denna gång med 1 miljon items istället

Den här gången blev resultatet ca 0.4 sekunder. Utifrån detta kunde vi konstatera att det skulle vara rimligt att implementera en gräns på hur många items man kan plocka upp vid ett tillfälle på ungefär 1 miljon items

Byggscript före

```
<project name="Projekt Build Script" default="TestSuite">
    <description>
        Ett Ant script som kompilerar src & test koden
    </description>

    <!-- Kompilerar .java filer i mappen "src"-->
    <target name="CompileSrc">
        <javac srcdir="src" destdir="bin/main"/>
    </target>

    <!-- Kompilerar .java filer i mappen "tests"-->
    <target name="CompileTests" depends="CompileSrc">
        <javac srcdir="tests" destdir="bin/test">
            <classpath>
                <pathelement location="lib/junit-4.12.jar"/>
                <pathelement location="lib/hamcrest-core-1.3.jar"/>
                <pathelement location="bin/main"/>
            </classpath>
        </javac>
    </target>

    <!-- Does amazing stuff, trust me on this! -->
    <target name="TestSuite" depends="CompileTests">
        <junit printsummary="on" haltonfailure="no" fork="yes" showoutput="true">
            <classpath>
                <path id="classpath.test"/>
                <pathelement location="lib/junit-4.12.jar"/>
                <pathelement location="lib/hamcrest-core-1.3.jar"/>
                <pathelement location="bin/test"/>
                <pathelement location="bin/main"/>
            </classpath>
            <formatter type="brief" usefile="false"/>
        </junit>
    </target>
</project>
```

Byggscript slutliga

```
<project name="Projekt Build Script" default="TestSuite">
    <description>
        Ett Ant script som kompilerar src & test koden samt kör testsviten
    </description>

    <!-- Kompilerar .java filer i mappen "src"-->
    <target name="CompileSrc">
        <javac srcdir="src" destdir="bin/main"/>
    </target>

    <!-- Kompilerar .java filer i mappen "tests"-->
    <target name="CompileTests" depends="CompileSrc">
        <javac srcdir="tests" destdir="bin/test">
            <classpath>
                <pathelement location="lib/junit-4.12.jar"/>
                <pathelement location="lib/hamcrest-core-1.3.jar"/>
                <pathelement location="bin/main"/>
            </classpath>
        </javac>
    </target>

    <!-- Kör testsviten och skriver ut outputen till en loggfil and does amazing stuff to! -->
    <target name="TestSuite" depends="CompileTests">
        <record name="logfile.txt" action="start" append="false"/>
        <junit printsummary="on" haltonfailure="no" fork="yes" showoutput="true">
            <classpath>
                <path id="classpath.test"/>
                <pathelement location="lib/junit-4.12.jar"/>
                <pathelement location="lib/hamcrest-core-1.3.jar"/>
                <pathelement location="bin/test"/>
                <pathelement location="bin/main"/>
            </classpath>
            <batchtest>
                <fileset dir="tests">
                    <include name="AllTests.java"/>
                </fileset>
            </batchtest>
            <formatter type="brief" usefile="false"/>
        </junit>
        <record name="logfile.txt" action="stop"/>
    </target>
</project>
```