Kommentarer World of Zuul

Jag tycker du har gjort ett bra jobb med spelet. Det var lätt att se vilka kommandon fanns och vart man kunde gå. Även bra med felmeddelanden och liknande.

Det finns detaljer som kanske borde ändras. Jag ger förslag och varför man skulle kunna skriva på ett annan sätt men inget av dem är ett måste utom möjligen sättet du ser ifall personen är i rätt rum. Mer om det finns i nästa stycke.

Hårdkodade jämförelser

För till fället ser du ifall personen är i ett visst rum genom att använda den *description* som rummet har. Det kan bli jobbigt ifall man ändrar på beskrivningen på ett rum för då kommer resten av koden helt plötsligt tycka att det är ett helt annan rum. Exempel på kod från ditt spel:

```
1 currentRoom.getShortDescription().equals("in the town's harbor")
```

Mitt förslag är att jämföra *currentRoom* med *harbor* istället, så att den kollar ifall det är samma rum snarare än att kolla ifall beskrivningen är samma. För att det ska fungera måste du göra så att man kan komma åt rummen. För tillfället är de lokala variabler i *createRoomsAndItems*() men du skulle kunna flytta ut dem och ha dem som instancevariabler istället:

```
1 public class Game {
2     private Room townSquare, giftShop, fleastinkHotel, room605, darkAlley
          , harbor, submarine, pub, abandonedFactory, sewers, airport;
3     ...
```

och sedan ett exempel på en jämförelse:

```
1 currentRoom.equals(harbor)
```

Det finns liknande problem med föremål och väderstäck. Genom att använda text som sätt att hämta ut föremål eller dörrartill andra rum kommer inte kompilator att upptäcka fel. För att lösa det kan man använda sig av *enums* eller konstanter. Ifall du då råkar skriva $NOTRH$ istället för $NORTH$ kommer kompilatorn ge ett felmeddelande och du slipper spela igenom hela spelet bara för att hitta felet. Det kan vara lite överkurs så det kan skippas. Jag ger ändå ett exempel på hur det skulle kunna se ut:

```
1  public class Game {
2      private static enum Directions {
3          NORTH("north"), SOUTH("south"), WEST("west"), EAST("east");
4          Directions(String value) {
5              this.value = value;
6          }
7          private String value;
8          public String getValue() {
9              return value;
10         }
```

```
11      }
12      ...
13      private void createRoomsAndItems() {
14          townSquare.setExit(Directions.NORTH.getValue(), fleastinkHotel);
15      }
16      ...
17  }
```

eller med hjälp av konstanter:

```
1   public class Game {
2       private final static String NORTH = "north";
3       private final static String SOUTH = "south";
4       ...
5       private void createRoomsAndItems() {
6           townSquare.setExit(NORTH, fleastinkHotel);
7       }
8       ...
9   }
```

Kommentarer

Du har kommentarer på både svenska och engelska. På de flesta ställerna är det för att hjälpa mig att förstå var du har ändrat men i t. ex: *Player* står en del på svenska och del på engelska eller så har du helt hoppat över att skriva en riktigt kommentar.

Du har hoppat över att beskriva in och utdata på några funktioner. Det kan vara överflödigt att ha på en del eftersom variabelnamnen är rätt självförklarande.

Det finns egentligen inte så stor anledning att skriva långa dokumentationskommentarer på privata metoder. De kommer aldrig att synas i dokumentationen och ska därför bara förklara ifall något skulle vara svårt att förstå i metoden. Det här är bara ett tips ifall man tycker det är tråkigt att skriva kommentarer och tycker egentligen inte att du ska ändra något i koden.

Klamrar

På många ställen i koden står det:

```
1   if (condition) {
2       //Do something
3   }
4   else if (condition) {
5       ...
```

Det vanligaste sättet är att flytta upp *else if* till raden åvan så det blir:

```
1   if (condition) {
2       //Do something
3   } else if (condition) {
4       ...
```

och på klasser och metod namn:

```
public class Game
{
    ...
//Borde vara
public class Game {
    ...
```

Hela det här stycket är också bara tips. Ingen av sättet är rätt eller fel men man bör vara konsekvent med hur man gör det. Bluej krånglar också till det genom att lägga till klamrar på fel sätt.