UML for Sprint 1

Boxes and arrows and shit
UML For the Sprint 1
UML For the Sprint 1 - The Room class
UML For the Sprint 1 - Room

Class name

variables

constructors and methods

inner class (not orthodox)
UML For the Sprint 1 - Room

Class name: se.itu.game.cave.Room

variables:

visibility (, -, #, +) name : Type

-north : Room

constructors and methods:

visibility name(arguments) : Type

+removeThing(thing : Thing) : Thing
UML For the Sprint 1 - Room - Lists of stuff

-things : Thing [*]

or:

-things : Thing [0..*]

Could be a java.util.List<Thing> for instance
<<enumeration>>

Room.Direction

In Java:

class Room
{
    public enum Direction {
        NORTH, EAST, SOUTH, WEST
    }
}

```java
public enum Direction {
    NORTH, EAST, SOUTH, WEST
}
```
UML for Player

<<singleton>>

-instance : Player

New thing! Static - underline
Associations

- Player knows its current Room
- Player has 0..* Thing
- Room has four other Room:s
- Room has 0..* Thing

For instance:

Player<>------>Room

1
Composition vs Association vs Aggregation

**Composition** - “is part of” relationship - an object “owns” another object (which is part of the first object). If an object creates another object - it “owns it” - when the object dies, the other object dies too.

**Aggregation** - “has a”, “knows”, “uses” - an object uses another object.

- The Room doesn’t own the four Room:s in N, S, E, W - but it can use them.
- The Player doesn’t own the currentRoom - but can use it (look for things etc)

In all cases, variables are used.
Inheritance

Implementation inheritance (implementing an interface) uses a dotted arrow from the implementing class to the interface.

Inheritance using *extends* uses a normal arrow from the extending class to the super class.

In the UML for Sprint 1, we don’t use inheritance.