# OOP Week 6 Exercises

# Inheritance and Polymorphism

**Learning Outcomes**

From this session, you will be able to:

* Understand inheritance and how to implement it in java
* Use super classes and subclasses
* Write and use overriding and overloading methods

## Exercise 1 – Implement the Entity class

We are going to continue working with the Entity and World classes designed and implemented in week 4. We want to practise inheritance so we are creating a superclass Entity that encapsulates all the main features of a general Entity.

The information that we need to store about a general entity is its **name**, position as coordinates **x** and **y**, the **world** that the entity belongs to (We will add it later) and its **symbol**.

* Create a new project in NetBeans and implement or copy the class **Entity** with those attributes, a constructor, and the getter, setters.
* Check that the class work by creating an entity in your **main program**, set the coordinates **x** and **y** and displaying the information about its position.

## Exercise 2 – Implement the Hobbit class

Implement the class **Hobbit** as a subclass of **Entity**. A Hobbit is represented by the symbol # and an int attribute health.

## Exercise 3 – Implement the Wizard class

Implement the class **Wizard**. The wizard’s symbol is @, and it has age and wisdom attributes. Implement the constructor that initialises the name, age and wisdom attributes from parameters.

## Exercise 4 – Overriding move method

Every Entity can move, add the method move to the Entity class. This method does not anything in that class. Now you can implement the method move for Hobbit and Wizard.

* Hobbits can only move to a random position adjacent where they are and every time that they move his health is decreased by one.
* A young wizard (age < 100) moves to any position (x and y can be a random number between 0 and 100) and older wizard cannot go further than 5 positions where they are (*Hints: add a random number between -5 and 5 to the x and y position*)

## Exercise 5 – Overriding toString method

Every class inherited from the Object superclass the method **toString**, which returns a String with useful information about that class. Implement a **toString** method for each class.

* The method **toString** for the **Entity** class returns a String of the following form:

**name** **symbol** position **x** **y**

* The method **toString** for the **Hobbit** class returns a String of the following form:

same information as the Entity class + health **health**

* The method **toString** for **Wizard** returns:

same information as the shape class + wisdom **wisdom** and age **age**

Check how the methods work in the main program by invoking these methods and displaying the returned strings.

## Exercise 6 – Overload constructors

We want to have two ways to create instances of hobbits and wizards. We want to create hobbit with a value of **200** by default and wizards with a value **100** for wisdom by default (with **age** still specified by parameter). Write a **second** **constructor** for these two classes. We will have now overloaded their constructors, so that there are two ways to create instances of the classes.

## Exercise 7 - Abstract Classes

We have decided to convert the **Entity** class into an **abstract** class.

* You have to convert the **Entity** class into an **abstract** class
* Convert the **move** method into an **abstract** method.

## Exercise 8 - World Class

We are now going to add the class **World** implemented in week 4 to our project. The class World has a name and stores an **ArrayList** of **Entitys**.

* Add the attributes to the class
* Implement a constructor that receives as parameter the name of the world
* Implement a method **addEntity(Entity entity)** to add an Entity to the list.
* Implement a method **deleteEntity(int pos)**  that given a position in the list, removes the entity in that position
* Implement a method **getEntity(int pos)**  that returns the entity in that position
* Add the attribute **world** to the **Entity** class with the corresponding getters and setters