Tutorial 4: Inheritance

CSE1100 - Introduction to Programming

1 Basic Inheritance

Create a class `Building`. A building has a `String street` and an `int value`. Make sure you encapsulate these properly. A building's street should never be able to change, but a building's value can change. Also, give the class a constructor that initialises all fields.

Create a subclass of `Building`: `House`. A house should have an `int number` with a getter. Give this class constructor that initialises all fields too.

Create another subclass of `Building`: `Office`. An office has an `int amountOfWorkers` with a getter and a setter. Give this class a constructor that initialises all the fields.

2 Distribution of Logic

You are given three classes `Person`, `Student` and `Teacher`. Inspect these classes. You can see there is a lot of code duplication. Rewrite these classes to use inheritance and to minimilise code duplication.

Implement the `toString` methods for all three classes. `toString` should return a `String` with *all* properties in a human readable way. It should also be clear from `toString` on which class it is being called. An example `toString` output might be:

```
Student:
Thomas is 1.9 metres tall and lives in Delft
```

3 Interface

Copy your `Person`, `Student` and `Teacher` classes from 'part 2: Distribution of Logic'. You are given an interface `HasToStudy`. Make `Student` implement `HasToStudy` in the following way:

- The `Student` keeps track of how many times `study` is called.
- If and only if `study` has been called at least 5 times, `willPassExam` should return true.

3.1 Testing

Test the `Student` class. Write a test where `willPassExam` returns `true` and a where `willPassExam` returns `false`. Write at least one test for every other method as well.

4 Equals and Hashcode

Copy your `Person`, `Student` and `Teacher` classes from 'part 3: Interface'. Implement a proper `equals` and `hashCode` method for `Person`, `Student`, and `Teacher`. The `equals` and `hashCode` methods should use *all* available fields, e.g. a `Student` is only equal to another if their name, height, living status, and amount of times studied are the same. Make sure to minimalise code duplication.

4.1 Testing

Write a test for each `hashCode` method and three tests for each `equals` method:

- One test that checks equality for two objects at the same memory location.
- One test that checks equality for two objects that are equal but not at the same location in memory.
- One test that checks equality for two objects that are not equal.