Week 2 Lectures

Subclass and inheritance

In the lecture, we talked about how to implement a graphical circle and there are basically 3 ways to do so. First approach is basically just brute force, you implement everything specifically to graphical circle. The second approach and third approach are what we are interested here.

IS-A relationship vs HAS-A relationship.

IS-A relationship is inheritance. For example, A kiwi is a fruit with hairy skin, oak tree in my yard is a tree. The concept of a class is useful because of its reusability. One important thing to remember is that the derived class is an instance of the base class.

HAS-A relationship is containment. For example, you might create a Business class that contains an array of Department objects. You would not say, “A department is a business,” but you would say, “A business has departments.” Therefore, this relationship is not inheritance; it is a form of containment called composition —the relationship in which a class contains one or more members of another class, when those members would not continue to exist without the object that contains them. (For example, if a Business closes, its Departments do too.)

Similarly, each Department object might contain an array of Employee objects. In this case, you would not say, “An employee is a department,” but you would say “A department has employees.” This relationship is not inheritance either; it is a specific type of containment known as aggregation —the relationship in which a class contains one or more members of another class, when those members would continue to exist without the object that contains them. (For example, if a business or department closed, the employees would continue to exist.) On the other hand, if Employee s no longer existed, no EmployeeWithTerritory would exist either.

toString(), equals() and hashCode()

- toString, you can customise your print statements
- equals is probably the most important one. read the link.
- hashCode is used to hash things in your class, i.e. username, password. You can override the original algo

Abstract class

- similar to header file in C, it is a class with declared methods
- cannot be instantiated

multiple inheritance

- a subclass can only extend one parent class
- you need to use interface

Interface

- variables declared must be static and final
- a class can implement more than one interfaces

```
// light is wave and particle
public class Wave{

}

public interface Particle{

}

public interface Love{

}

public class Light extends Wave implements Particle, Love{

}
```

Please note that interface can have methods

```
default public boolean canPickUp(Entity e, Player player){
    boolean playerHasKey = player.getInventoryItems().stream()
        .anyMatch(entity -> entity.getType().equals("key"));
    boolean isKey = (e instanceof Key && !(e instanceof SunStone));
    return isKey;
}
```

TODO: https://gitlab.cse.unsw.edu.au/COMP2511/22T2/content/-/blob/master/lecture-slides/week02/pdfs/OOP_inJava_Poly_Rest.pdf

Tutorial

Activity A

In groups, analyse the classes to answer the following questions:

1. What is the difference between super and this?

- super: The instance of the super class
- this: The instance of this class (like self in python)

2. What about super(...) and this(...)?

- super(...) Runs the respective method in the super class
- this(...) Runs the respective method in this class with the given parameters (used for method overloading)

3. What are static fields and methods?

- static: the particular member belongs to a type itself, rather than to an instance of that type.

Activity B

Within the src directory, create a new package called employee.

Create an Employee class which has private fields for an employee's name and salary and appropriate getters, setters, and constructors. Document the class with JavaDoc.

Use VSCode to create the getters and setters. In this course we are not going to require that you write JavaDoc, except when specified.

1. What is meant by the term "self-documenting code"?

meaningful function name and variable name, readable code

2. Explain why comments can be considered a code smell.

- comments are typically not updated and they can become irrelevant and misleading
- It could be argued that the requirement for a comment means the code is not self-documenting
- don't comment unless you really really have to (style mark)

3. Discuss as a class whether code should have comments / JavaDoc

Activity C

This exercise continues on from the Employee class in Exercise B.

- How many constructors should the class have? What arguments should they take?

Create a Manager class that is a subclass of Employee and has a field for the manager's hire date.

- What constructors are appropriate?
- Is appropriate to have a getter for the hire date? What about a setter?

Override the toString() method inherited from Object in both classes.

- What should the result of toString() contain?
- Does the method in Manager call the one in Employee?