Class 2: Inheritance

COMP8710

1 Introduction

These exercises are about inheritance.

2 Polymorphic types

Solutions for the following questions will become available on moodle after the end of the PC class.

Assume that you have four classes: Person, Instructor, Student and PhDStudent. Instructor and Student are both subclasses of Person. PhDStudent is a subclass of Student.

Which of the following assignments are legal and why?

```
Person p1 = new Student();
Person p2 = new PhDStudent();
PhDStudent phd1 = new Student();
Instructor i1 = new Person();
Student s1 = new PhDStudent();

s1 = p1;
s1 = p2;
p1 = s1;
i1 = s1;
s1 = phd1;
phd1 = s1;
```

3 An inheritance hierarchy

Assume that you have four classes - O, X, T and M - and a variable of each:

```
O o;
X x;
T t;
M m;
```

Assume that the following assignments are all legal:

```
m = t;
m = x;
o = t;
```

Assume that the following assignments are all illegal:

```
o = m;
o = x;
x = o;
```

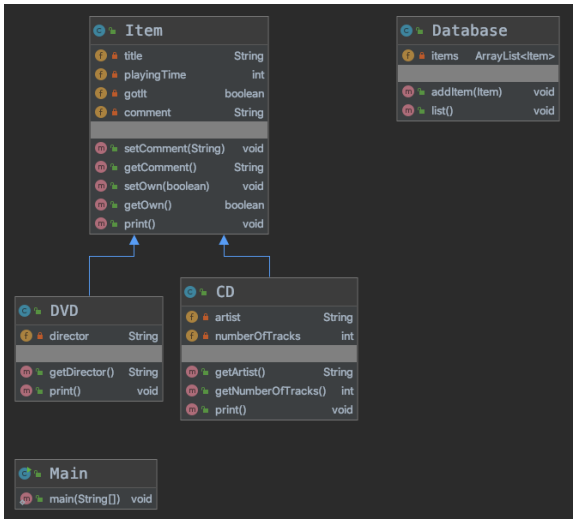
What can you say about the relationships of these classes; i.e., which classes are a subclass or a superclass of which others?

4 Method overriding

Download and open the `dome-v3` project from `moodle`.

1. Which method in the CD class overrides a method in the Item superclass? Within IntelliJ there are a couple of tools that can help you explore inheritance relationships:

   (a) **Class Diagram**: Select all the java classes in the project navigation pane. Right click on the selection and select `Diagrams -> Show Diagram`. In the new window, enable the visualisation of `fields` and `methods` from the toolbar (select "f" and "m" buttons). You should see a diagram similar to the one below:



   (b) **Editor hints**: Open the CD class and navigate near the signature of the `print` method. Look at the margin space left of the method name. What do you see? Hover the mouse over the icon. What happens if you click on it?

2. Within the `Main.main()` method, create a new CD object and call the method print. Run your program. Can you explain the output of your program?

3. (This exercise is slightly artificial but it makes an important point.) Change the header of the `print` method in the CD class so that it takes a single `int` parameter whose value is to be printed out when the method is called. Add the extra print statement to the body of the method. Recompile and run your program again. What difference has adding the parameter made to what you see in the output of your program? Can you explain this?

4. Remove the parameter and the extra statement you added in the previous exercise. Now add the following statement to the body of the print method in CD:

   ```
   super.print();
   ```

   What difference do you think that adding this statement will make when a CD's print method is called in future? Test out your prediction.

   Move the call to super.print to the end of the method and predict what the output will be. Try it out. Were you right?

5 Dynamic type vs static type

1. Open the `Database` class of the dome-v3 project. Find its `list` method. What is the type of *the variable* that is used to call the print method? We call that the *static type* of a variable.
2. When print method in list is called, what type of object will it be called on? (Hint: there are two possibilities.) We call that the *dynamic type* of a variable.
3. From what you can see of the code in the list method, and what you have seen from the earlier exercises, predict what would be printed by the list method if you added a single CD to the database. Try it out to see if you are right: In the main method create a Database object, and add the CD object you have.

6 The toString method

1. All classes inherit from a class called `Object`. That class defines the following method:

   ```
   public String toString()
   ```

   In the `main()` method create a new DVD object. After creating it, type the name of the variable followed by the dot, and wait for IntelliJ to show you a list of methods that you can call. Scroll through the list, and you will find the method `toString` it inherits from `Object`. Call it and print the value. What string does it return? Does it contain any useful information about the DVD object you created?
2. Adjust the code for the `System.out.println()` call you have created so that you just pass the DVD object, wihtout calling the method `toString()`. Run your program. Notice that the `println()` method internally uses the `toString()` method to generate the output that should be printed for that object.
3. Override the toString method in `DVD` class so that it returns the director as its result. IntelliJ allows you to quickly generate the correct method signature when you try to override a method. Open the class DVD, select from the menu `Code->Override Methods...`. Select the method you want to override, and click OK. A skeleton version of the method is automatically generated. Modify the code to return the value that you want. Check the icon next to the method signature to make sure you have properly overridden the inherited version. Run your program and see the output of your code now.
4. Add a definition of `toString` in the `Item` class that returns the `title` of the item. Note that it is this version of toString that now overrides the one defined in `Object`, and the one you defined in `DVD` overrides the one defined in `Item`.
5. Modify the `toString` method in `DVD`, and make it call the `toString` method in `Item` so that it can return a string that contains both the `title` and the `director`.
6. Replace the print methods in Item, CD and DVD with toString methods that return the strings that would have been printed. You will also need to change the list method of Database.

END OF CLASS