

Lab 6 – Object Inheritance

Aim

This week is focused on object inheritance. This builds on your object work from week 5, and extends it using the inheritance and overriding discussed in lecture 6.

Tips:

1. Objects are a very important Java concept, so it is important you understand what you are doing.
2. Complete the previous lab sheet before you start this sheet.
2. There is a checkpoint near the beginning of this lab sheet. However, it is important that you continue with the sheet after the checkpoint.

Extending Person to Student

In lab 5, you created a Person object. However, we also want to create a new Student object. This Student will inherit all the methods and attributes from Person, and add extra attributes for major and year of study.

- The student object will have 2 new instance variables
 - `String major;`
 - `int yearOfStudy;`
- Your project will have 3 classes
 - `Week6Checkpoint.java` : your main class, the controller. **This class will be the only class with a main method.**
 - `Person.java`: your Person class from last week
 - `Student.java`: this class must inherit from person, and add the two new properties.

Part one: Simple Inheritance

Create a new Java project, named `Week6Checkpoint`, with a class with a main method. Add your person class from last week, and ensure that this has no main method.

- Create a student class that extends Person
- This student should have the 2 student specific variables, major and yearOfStudy
- Create a constructor that receives all of the arguments needed to create a Student

```
Student student1 = new Student("Ting", 22, "Computing Science", 123, "ICT", 2);
```
- The student constructor should also call the person constructor to set the Person specific attributes
 - HINT, you will need to call `super()` (see lecture 6!)
- Add getters and setters for encapsulation

Part two: Creating and Using your Student

Now that a student has been created, you can use the methods in the Student class, but you can also use the methods in the Person class (such as the birthday method), without needing to create additional methods.

- You can now call your birthday method from the Student object, but **without** creating a birthday method in the Student class! Call it.
- Use your getter methods to get the age and major of your new student and display them.

```
Output - Lab6CheckpointIdea (run)
run:
Person called Ting created
Happy Birthday Ting!!!
You were 22 but you are now 23
Major is ICT
BUILD SUCCESSFUL (total time: 0 secc
```



Checkpoint

Show a TA your student object, extending a person object.

Answer any questions they have, and then **continue with the sheet**

This will count towards your final grade.

In the Cake Shop - Cakes

The CakeShop is a project that will simulate a cake shop. Firstly, the cakeshop will be able to create cakes, as we did in the lectures. However, we can also create wedding cakes, which inherit from the cake objects.

- Download the Netbeans project folder from ICE, and open it in Netbeans. Open Week6Cake. You should understand what the code is doing. If not, ask!
- Add this line of code to the constructor:

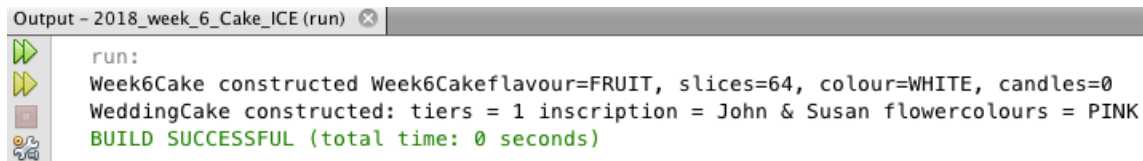
```
System.out.println("Week6Cake constructed "+this.toString());
```

- Now open CakeShop. **This is the controller class which has the main method.** In main, create a Week6Cake.
- Look at the console output: is it what you expect?
- Add an .equals() method to the Week6Cake class. Test it. (see lecture notes).

In the Cake Shop - WeddingCakes

As discussed in the week 6 lecture, inheritance allows objects to be created that inherit the attributes and methods of an object, and extend them. We will extend our Week6Cake to create a Wedding Cake object.

- Open WeddingCake. Look at the code. You should understand what the code is doing. If not, ask!
- Then go back to CakeShop, and in the main method, comment out the Week6Cake object, and create a WeddingCake object instead.
- Look at the printout. You should see something like this:



```

Output - 2018_week_6_Cake_ICE (run)
run:
Week6Cake constructed Week6Cakeflavour=FRUIT, slices=64, colour=WHITE, candles=0
WeddingCake constructed: tiers = 1 inscription = John & Susan flowercolours = PINK
BUILD SUCCESSFUL (total time: 0 seconds)

```

- Both a Week6Cake and a WeddingCake were constructed. Do you understand why? If not, ask!

Cake Cutting

The cake shop also want to create a cutting plan. They want the user to input how many slices they need, and generate a 'slice plan' showing how to cut the cake to get that number of slices.

- Open CakeCutPlan. Look carefully at the object instance variables and the constructor. The comments should help you. If you don't know what the code does, ask!
- In the constructor, initialise the variables cakeX, cakeY, cakeWidth, cakeHeight. Suggested values: 40, 40, 720, 720.
- Go back to CakeShop. Create a WeddingCake, and then create a new CakeCutPlan object, passing the Cake as an argument:

```

WeddingCake weddingCake = new WeddingCake(1, "John & Susan", "PINK", "FRUIT", 64, "WHITE");
CakeCutPlan ccp = new CakeCutPlan(weddingCake);

```

- You should see a window displaying your wedding cake!
- You can change the image file in the project folder to your own cake image; change the name in the loadImage() method in the constructor

```

//make sure you have an image file in your project folder
image = loadImage("cake.jpg"); //this must be the name of your image file EXACTLY

```

Cake Cut Plan

The final stage is to display how a cake is to be cut. The idea is that given a desired number of slices, it will show how to cut the cake to the correct size



- Follow the code, starting in the main method in Cakeshop.
- In the constructors for Week6Cake and WeddingCake, a number of slices is set. This cake object is then passed to CakeCutPlan as an argument to the constructor.
- In the CakeCutPlan constructor, the number of slices is received from the cake object like this:
`int numSlices = cake.getSlices();`
- There is also a paint() method in the CakeCutPlan, which will draw a line on the screen. You are going to change it to draw multiple additional slices.
- Go to the constructor of CakeCutPlan. Using the number of slices in the Cake object, you need to calculate the sliceWidth and sliceHeight values. You can assume that both the cake and the slices are square (i.e. a cake with 16 slices is 4 by 4).
- HINT - You previously set the cakeWidth and cakeHeight variables in a previous stage (720), and a slice width is therefore going to be narrower than 720...How would you calculate it?
- Add a system.out.println of the calculated values. Do they look OK? Change the number of slices in the constructor (in main in CakeShop). Check again. Note: only use square numbers here: 16, 36, 64 etc.
- If these numbers look OK, move down to the paint() method. We are **overriding** this method to do our painting.

```
g.drawRect(cakeX, cakeY, cakeWidth, cakeHeight);
g.drawLine(cakeX+cakeWidth/2, cakeY, cakeX+cakeWidth/2, cakeY+cakeHeight);
//here you will need to create two for loops, one to draw the vertical lines,
//one to draw the horizontal lines
//g.drawLine(x1, y1, x2, y2);
}
```

- Currently, you will see that the cake is cut into 2 slices, using a single line of code in the paint method:
`g.drawLine(cakeX+cakeWidth/2, cakeY, cakeX+cakeWidth/2, cakeY+cakeHeight);`
- This draws a vertical line half way through the cake image.
- As shown in the lecture notes, you need to add two for loops, one to draw vertical lines, one for horizontal lines to show how to slice the cake, eg for 64 slices:

Next Steps – Homework 6

Objects are a big concept, so this week, there is no homework sheet. Instead, you should ensure you understand this code, and creating objects. Study your lecture notes from weeks 5 and 6 carefully, and make sure both lab sheets are fully completed.

- As an additional challenge, can you make a cake cut plan for round cakes?