COMP1102/8702 – Practical Class 4

A Complete Application with Objects Aims and Objectives

This laboratory has been designed to help you

- learn how to define classes which are instantiated in the main method of an application;
- gain further practice with defining methods which are only used within a class (object) and those which are intended to be invoked using an object reference.

Getting Started

Start *IntelliJ* and open the project called "Practical04" (download it from FLO).

Task 1

On completion of this task you will have demonstrated the ability to construct

- a simple class definition containing instance variables and methods, and
- a class containing only a *main* method which creates an instance of a class (an object) and accesses instance variables, and invokes methods, through a reference to the object.

In *IntelliJ* add the following to the files BoatMaker.java (the Java main class) and Boat.java (the Java class file):

1. BoatMaker – this class contains a main method which performs the following actions:

i. Prints the message "Starting boat application"

ii. Defines a variable called myBoat of type Boat and which is initialised to a new Boat object.

iii. Invokes (calls) the print method from myBoat

iv. Changes the value of the name attribute in myBoat to be "Fred"

v. Invokes the print method again, to illustrate the effect of the change

2. Boat – this class contains:

i. an instance variable, regNum, of type int initialised to -1 ii. an instance variable, bClass, of type String initialised to "unknown" iii. an instance variable name of type String which is initialised to "unknown"

iv. a method print which does not return a value (i.e. the return type is void), has no formal parameters and prints the boat's name, class and registration number. in the following format:

Boat *name*, Class = *bClass*, Registration # = KA *regNum*

where the words in *italics* should be replaced with the values of the corresponding instance variables.

Compile and run the application from *IntelliJ*. The following output should be produced:

```
Starting boat application
Boat unknown, Class = unknown, Registration # = KA -1
```

Checkpoint 17

Have the program source code and output marked by a demonstrator

Task 2

Modify the program developed in Task 1 in the following ways.

1. Add the following to the end of the main method contained in the class BoatMaker:
   i. set the instance variable regNum contained in the object referred to by myBoat to 6467.
   ii. set the instance variable bClass contained in myBoat to "International 505".
   iii. set the instance variable name contained in myBoat to "Harmony Blue" iv. invoke (call) the print method from myBoat
2. Compile and run the application from *IntelliJ*. The following output should be produced:

```
Starting boat application
Boat unknown, Class = unknown, Registration # = KA -1
                                   Boat Harmony Blue, Class = International 505, Registration # = KA 6467
```

Checkpoint 18

Have the program source code and output marked by a demonstrator

Task 3

Modify the program developed in Task 2 as directed below.

1. Extend the definition of the class Boat to include a constructor with the following formal parameters:

i. the_name of type String ii.

the_bClass of type String iii.

the_regNum of type int

The constructor should assign each of the class's instance variables to the respective formal parameter. For example, regNum should be assigned to the value of the_regNum. For example, new Boat("Harmony Blue","International 505",6467); constructs a new Boat object.

Note that

- constructors do not have a return type and
- defining your own constructor, effectively undefines the default constructor (Boat()).

2. Extend the definition of the class Boat to include a default constructor, namely,

   Boat() {}

3. Modify the main method in the definition of the class BoatMaker to perform only the following:
   i. Print the message “Starting boat application”
   ii. Define a variable called myBoat1 of type Boat and initialise it to a new Boat object using the default constructor (as in Tasks 1 and 2).
   iii. Define a variable called myBoat2 of type Boat and initialise it to a new Boat object constructed with the actual parameters "Harmony Blue", "International 505" and 6467.
   iv. Invoke (call) the print method from myBoat1 v. Invoke (call) the print method from myBoat2

4. Compile and run the application from *IntelliJ*. The following output should be produced:

```
Starting boat application
Boat unknown, Class = unknown, Registration # = KA -1
                    Boat Harmony Blue, Class = International 505, Registration # = KA 6467
```

Checkpoint 19

Have the program source code and output marked by a demonstrator

Task 4

Modify the program developed in task 3 as directed below.

1. The class Boat should be extended/modified to include the following:

   i a constant called yardstick with a value of 95.0, ii all instance variables should be private,

   iii a method called setName which sets the instance variable name to a value which is passed as a parameter, iv a method called getName which returns the value of the instance variable name.

2. Add statements to the end of the main method to do the following:
   i Change the name of myBoat1 to "Australia II" (invoke the setName method in the object referred to by myBoat1).
   ii Invoke (call) the print method from myBoat1.
   iii Print the names of myBoat1 and myBoat2 (invoke the getName method in both of the objects referred to by myBoat1 and myBoat2 and print out the results).

3. Compile and run the application from *IntelliJ*. The following output should be produced:

```
Starting boat application
Boat unknown, Class = unknown, Registration # = KA -1
Boat Harmony Blue, Class = International 505, Registration # = KA 6467
Boat Australia II, Class = unknown, Registration # = KA -1
Australia II
Harmony Blue
```

Checkpoint 20

Have the program source code and output marked by a demonstrator

Task 5 (Extension Practice)

Modify the program developed in Task 4 as directed below.

1. The class Boat should be extended/modified to include the following:

i a count of the total number of Boat objects created. This should be incremented by 1 each time a Boat is created.

ii a static method called printTotal which outputs the total number of boats created. For example,

Total number of boats created = 4 iii the sequence number (incremented by 1 each time) of each boat should be stored and output with the other boat information as part of the Boat class's print method. For example,

Boat Harmony Blue, Class = International 505, Registration # = KA 6467
Sequence number = 3

iv the non-default constructor should be extended to include a formal parameter, previousBoat, of type Boat, the value of which should be stored in the newly created Boat object, by means of an instance variable of type Boat.

2. The main method should:

i Print the message "Starting boat application" ii Invoke the printTotal method.

iii Define a variable called boat1 initialised to a Boat object constructed with the name "B1", the class "C", the registration number 1000 and a boat (the forth argument to the constructor) of null (a special value meaning no object).

iv Define a variable called boat2 initialised to a Boat object constructed with the name "B2", the class "C", the registration number 1001 and a boat of boat1.

v Similarly define instance variables boat3 and boat4 so that each is constructed with a reference to the previous one.

vi Make use of a while loop which starts with boat4, invokes its print method and then does the same for the boat stored by its constructor (previousBoat) and so on until all boats have been printed. The code **must not** refer to the variables boat1, boat2 or boat3.

vii Invoke the printTotal method.

3. Compile and run the application from Grasp. The following output should be produced:

```
Starting boat application
Total number of boats created = 0
Boat B4, Class = C, Registration # = KA 1003
Sequence number = 4
Boat B3, Class = C, Registration # = KA 1002
Sequence number = 3
Boat B2, Class = C, Registration # = KA 1001
Sequence number = 2
Boat B1, Class = C, Registration # = KA 1000
Sequence number = 1
Total number of boats created = 4
```