Peer Assisted Study Session

FIT2099 - Week 5

Monash University

Objectives

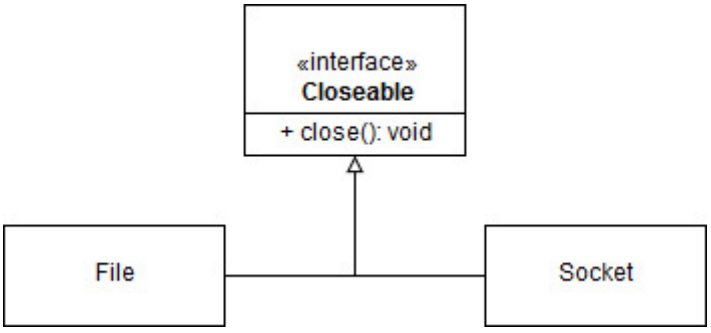
- Be able to draw a model from a given set of requirements
- Identify the use of abstract classes
- Understand upcasting and downcasting

Estimated Time

Question 1 - Question 2 (10 Minutes)
Question 3 - Question 4 (15 Minutes)
Question 5 (5 Minutes)
Question 6 - Question 7 (15 Minutes)

Questions

1. Consider a system library that provides classes for operations on Files and Sockets. Both these resources share some behaviors in common. For example, both of them can be closed when all processing is done. Model this simple commonality using either interfaces or abstract classes.



2. Explain your choice.

   Since we only wish to group common behaviors (and not attributes), the better choice is to create an interface to share behaviors between the two classes.

3. Create a base class with an abstract print( ) method that is overridden in a derived class. The overridden version of the method prints the value of an int variable defined in the derived class. At the point of definition of this variable, give it a nonzero value. In the base-class constructor, call this method. In main( ), create an object of the derived type, and then call its print( ) method.

```
abstract class Base {
 Base() {
   print();
 }
  abstract void print();
}

class Derived extends Base {
 int x = 5;
  void print() {
   System.out.printf("Derived.x = %d\n", x);
 }
}

class Main {
 public static void main(String[] args) {
   Derived d = new Derived();
   d.print();
 }
}
```

4. What will be the output when you run the main function?Explain why you get that output?

Derived.x = 0
Derived.x = 5

The constructor runs first so the constructor in the base class calls the print function before the variable x is defined, so it returns 0.

5. Read the following pseudo-java and find the code smell(s) it is exhibiting and identify the locations of the code smell(s):

```
public class Animal{}
public class Dog extends Animal{}

public static void main(String args[]) {
   Animal scooby = new Dog();
   Animal doo = (Animal)scooby;
}
```

Upcasting

6. Create an abstract class with no methods. Derive a class and add a method. Create a static method in the Main class that takes a reference to the base class, downcasts it to the derived class, and calls the method. In main(), demonstrate that it works. Enter the code below.

```
abstract class Empty {

}

class NonEmpty extends Empty {
 public void meth() {
   System.out.println("NonEmpty.meth()");
 }
}

class Main {
 public static void run(Empty e){
   NonEmpty ne = (NonEmpty) e;
   ne.meth();
 }
 public static void main(String[] args) {
   Empty e = new NonEmpty();
   run(e);
 }
}
```

7. Explain a way to remove the use of downcast in the above program.

   Put the abstract declaration for the method in the base class, thus eliminating the need for the downcast.

```
abstract class Empty {
 abstract void meth();
}

class NonEmpty extends Empty {
 public void meth() {
   System.out.println("NonEmpty.meth()");
 }
}
class Main {

 public static void run(Empty e){
   e.meth();
 }

 public static void main(String[] args) {
   Empty e = new NonEmpty();
   run(e);
 }
}
```