Q. What are the different testing approaches available at unit level? [2 marks]

* **Equivalence testing:** The possible inputs are classified into equivalence classes and a test case is selected for each class.
* **Boundary testing:** Special case of equivalence where boundary of the equivalence classes are tested. (0, empty string, year 2000 or Y2K bug)
* **Path testing:** this white box testing technique identifies faults in the implementation of the component by exercising all possible paths through the code at least once.
* **State-based testing:** Compares the resulting state of the system with the expected state

Q. Among OOP and FP, which programming paradigm you will choose in the following circumstances? (Explain why in one-two lines) [3 marks]

1. if same functions have to performed on different data types?
2. If set of operations are fixed new things will come up as the application grows?
3. When multithreading has to be done?

Solution:

1. If you have **many different related data types** that share functionality and plan to create more in the future, you will probably need to make use of **inheritance**. This will let you create modular and reusable code. Go with **OOP**.
2. **OOP** is good when you have a fixed set of operation on things, and as your code evolves, you primarily add new things. This can be accomplished by adding new classes which implement existing methods, and the existing classes are left alone.
3. With **FP**, for a specific input, the output will always be the same, and the input is always immutable. There are no state changes. So, whenever you need to write **thread safe code**, go with FP.

Q. Differentiate final and finalize()? [2 marks]

Answer: Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.

Finalize is used to perform clean up processing just before object is garbage collected.

Final is a keyword. Finalize is a method.

Q. You have to pick one of the keywords form the given keyword pool to fill the blanks (blanks are numbered from 1 to 10) in the given program such a way that it produces the desired output, as given. Also, describe the reason to pick the keyword for each space in one-two lines. You may use a keyword multiple times. [0.5x10 = 5 marks]

|  |  |
| --- | --- |
| **Program:**  interface A{  ­­void meth1();  }  abstract class B{  void meth1(){  System.out.println("Method 1 of abstract class");  }  \_\_**1**\_\_\_ void meth2();  \_\_**2**\_\_\_ void disp(){  System.out.println("From the abstract class");  }  }  class C \_\_**3**\_\_\_ B \_\_\_**4**\_\_\_ A{  int y;  C(int y){  this.y=y;  }  ­­­­\_\_**5**\_\_ void meth1(){  System.out.println("Method 1 of interface");  \_\_**6**\_\_.meth1();  }  void meth2(){  System.out.println("Method 2 of abstract class");  }  \_\_**7**\_\_ void disp(){  System.out.println("From class C,"+y);  }  }  class demo{  \_\_**8**\_\_ int z=20;  public static void main(String args[]){  A ob1=new C(10);  C ob2=new C(20);  B ob3=new \_\_**9**\_\_ ;  ob1.meth1();  \_**10**\_\_.disp();  ob3.disp();  System.out.println("From main class, z is :" +z);  }  } | **Keyword Pool:**  public,  private,  protected,  extends,  implements,  abstract,  final,  static,  super,  this,  A(),  B(),  C(),  ob1,  ob2,  ob3.  NONE: to describe nothing  OTHER: to describe extra keyword |
| **Desired output:**  Method 1 of interface  method 1 of abstract class  From class C, 20  From the abstract class  From main class, z is :20 |

Solution:

interface A{

void meth1();

}

abstract class B{

void meth1(){

System.out.println("Method 1 of abstract class");

}

abstract void meth2();

static void disp(){

System.out.println("From the abstract class");

}

}

class C extends B implements A{

int y;

C(int y){

this.y=y;

}

public void meth1(){

System.out.println("Method 1 of interface");

super.meth1();

}

void meth2(){

System.out.println("Method 2 of abstract class");

}

static void disp(){

System.out.println("From class C");

}

}

class demo{

static int z=20;

public static void main(String args[]){

A ob1=new C(10);

C ob2=new C(20);

B ob3=new C(30);

ob1.meth1();

ob2.disp();

ob3.disp();

System.out.println("From main class, Value of z is :" +z);

}

}

**(For 7th space those who have written none/static will be considered correct )**

Q. (a) What will be the output of the following program? Justify your answer. [2 marks]

class XYZ{

XYZ(){

System.out.println(“Constructor of XYZ”);

}

}

class PQR extends XYZ{

PQR() {

System.out.println(“Constructor of PQR”);

}

PQR(int a) {

System.out.println(“Constructor of PQR, value of a: ”+a);

}

}

class LNM extends PQR{

LNM() {

System.out.println(“Constructor of LNM”);

}

}

class DriverMain {

public static void main(String args[] ){

LNM lnmObject =new LNM();

}

}

Solution: Constructor of XYZ

Constructor of PQR, value of a: 10

Constructor of LNM

Q. What is singleton class concept? How it can be designed? Also, explain the practical life example where the singleton class concept will be used. [4 marks]

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time. After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created.

To design a singleton class:

1. Make constructor as private.
2. Write a static method that has return type object of this singleton class.

You can write code/pseudocode for singleton class describing the private constructor and static getinstance method.

public class MySingleTon {

private static MySingleTon myObj=null;

// Create private constructor

private MySingleTon(){

}

// Create a static method to get instance

public static MySingleTon getInstance(){

if(myObj == null){

myObj = new MySingleTon();

}

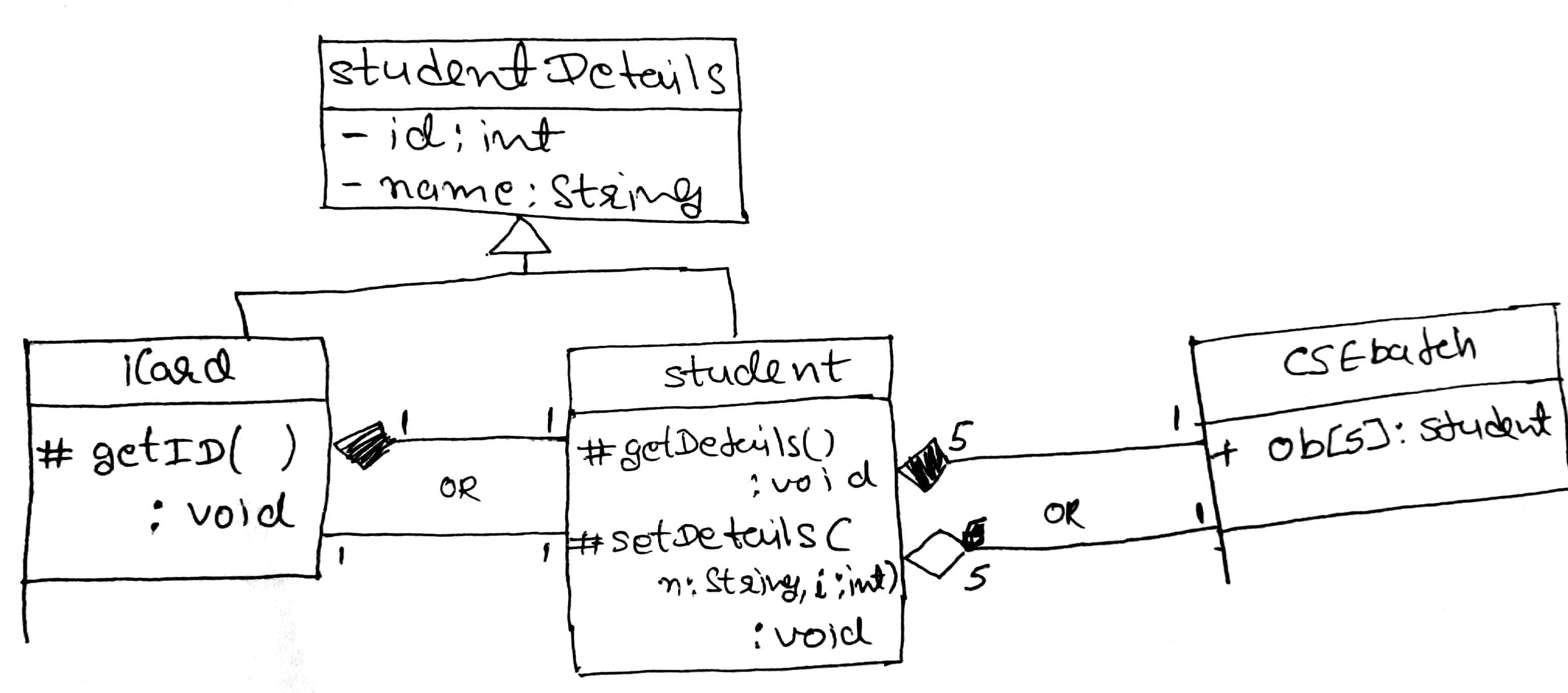
return myObj;

}

If multiple threads are accessing common resource which only have one access point, such as printer.

Q. Describe the following classes, attributes, relationships, and cardinality through the UML class diagrams? [3 marks]

|  |  |  |  |
| --- | --- | --- | --- |
| public class studentDetails{  private int id;  private String name;  } | public class student extends studentDetails{  protected void getDetails();  protected void setDetails(String n,  int i);  } | public class iCard extends studentDetails{  protected void getID();  } | public class CSEbatch{  public student ob[]=new student[5];  } |



Q. Assume you have written some classes and kept them into three packages, as listed in the following table. Classes access the members of other classes defined in other packages. [2 marks]

|  |  |
| --- | --- |
| **Package Name** | **Class Name** |
| LNMIIT.academic | department |
| LNMIIT.sports | resources |
| LNMIIT.managment | utilities |

1. Which lines of code will you need to add to each source file to put each class in the right package and to access another class?
2. How you can compile and run the class file department which is composed of main method?

Solution:

1. In the department class file following lines have to be added

Package LNMIIT.academic;

Import LNMIIT.sports.resources/\*;

Import LNMIIT.managment.utilities/\*;

And same in other class files

1. Either set a classpath or execute following commands from the parent directory (LNMIIT)
   1. To compile
      1. Javac academic/department.java
      2. Java academic/department

Q. in context of packages, how a name conflict can be resolved? Explain with an example.

Solution: using fully qualified name: Example:

If class A and class B have a method disp() with same name and signature and in another class from another package have imported both the classes then there would be a name conflict issue while calling the method disp(). As the compiler don’t know from which class the disp() method will be called.

Package MyPack1;

Class A{

Void disp(){ }

}

Package MyPack2;

Class A{

Void disp(){ }

}

//in different package

Package MyPack2;

Import MyPack1.\*;

Import MyPack1.\*;

Class C{

MyPack1.A ob1 new MyPack1.A();

MyPack2.A ob2 new MyPack2.A();

}