ASSESSMENT-2

1. Null pointer exception

2. <u>Exception:</u>

- Abnormal condition that occurs during the program execution.
- It will disrupt the normal program instructions.

Exception Handling:

- It is a mechanism to handle the exceptions

Exception handling ways:

Using keywords such as try, catch, throw, throws, finally

3. <u>Custom Exception:</u>

- Creates own exception

We write custom exceptions for,

- Business logic exceptions

4. <u>Encapsulations:</u>

- Wrapping the data, variables & methods

Rules:

- Class should be public
- Variable should be private
- Public method(Getter & Setter)
- Non-Parameterized Constructor(Default)
- Public properties

5. <u>Polymorphism:</u>

- Poly-Many
- Morph-form
- Same method name differ by no of parameters and return type

Types:

- Compile time polymorphism (Method overloading)
- Run time polymorphism (Method overriding)

6. Overloading:

- Same method name differ by no of parameters, type of parameters
- It is a Compile time polymorphism
- It increases the readability.
- We can create multiple methods in same class.

To achieve method overloading,

- Changing the no of parameter
- By changing the data type

Example: Addition of numbers

```
class Addition{
int add(int a, int b) {
int c=a+b;
}
float add (
float f1, floatf2)
float f=f1+f2;
}
}
```

7. Method Overriding:

In inheritance if subclass methods, super class method have same name with same parameter, same data type, same return type the sub class method will be executed & super class method will be overriding.

Example:

```
public class Rbi{
void interestrate(){
System.out.println("RBI Interest Rate is 12.5%");
}
public class Sbi extends Rbi{
@override
```

```
void interestrate(){
System.out.println(“RBI Interest Rate is 10.5%”);
}
}
public class Sib extends Rbi{
@override
void interestrate(){
System.out.println(“RBI Interest Rate is 8.5%”);
}
}
public class Test{
public static void main(String args[]){
Rbi rbi=new Rbi();
Sbi sbi=new Sbi();
Sib sib=new Sib();
rbi.interestrate();
sbi.interestrate();
sib.interestrate();
}
}
```

8. args

9. Error // Main method not found

10. Error // The parent class should be in interface

11. Abstraction:

It will hide the implementation details and show the functions only

There are 2 ways to achieve abstraction,

- Abstract class
- Interface

We can achieve 100% abstraction by interface

12. Initialize value:

Two ways to initialize the values,

- Implicit
- Explicit

Practical

```
public class MobileDetails {
	private String modelname;
	private int mobileid;
	private double price;
	private String memory;
	private String brandname;
	private String serialno;
	private int noofcamera;

	public String getModelname() {
		return modelname;
	}
	public void setModelname(String modelname) {
		this.modelname = modelname;
	}
	public int getMobileid() {
		return mobileid;
	}
	public void setMobileid(int mobileid) {
		this.mobileid = mobileid;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}
	public String getMemory() {
		return memory;
```

```
	}
	public void setMemory(String memory) {
		this.memory = memory;
	}
	public String getBrandname() {
		return brandname;
	}
	public void setBrandname(String brandname) {
		this.brandname = brandname;
	}
	public String getSerialno() {
		return serialno;
	}
	public void setSerialno(String serialno) {
		this.serialno = serialno;
	}
	public int getNoofcamera() {
		return noofcamera;
	}
	public void setNoofcamera(int noofcamera) {
		this.noofcamera = noofcamera;
	}
	@Override
	public String toString() {
		return "MobileDetails [modelname=" + modelname + ",
mobileid=" + mobileid + ", price=" + price + ", memory="
				+ memory + ", brandname=" + brandname + ",
serialno=" + serialno + ", noofcamera=" + noofcamera + "]";
	}
}

public class MobileDeatilsCRUD {

	private MobileDetails[] details = new MobileDetails[11];
	private int i = 0;

	public void getAll() {
		for (int i = 0; i < details.length; i++) {
			if (details[i] != null) {
				System.out.println(details[i]);
			}

		}
	}

	public String deletebyMobileId(int mobileid) {
		for (int i = 0; i < details.length; i++) {
			if (details != null) {
				if (mobileid == details[i].getMobileid()) {
					details[i] = null;
					return "Mobile Details Deleted: " +
mobileid;
				}
			}
```

```
		}
		return "Mobile Details Not Deleted";
	}


	public String updateBySerialNo(String oldName, String newName) {
		for (int i = 0; i < details.length; i++) {
			if (details != null) {
				if (details[i].getSerialno().equals(oldName))
{
					details[i].setSerialno(newName);
					return "Mobile Details Updated";
				}
			}

		}
		return "Mobile Details Not Updated";

	}
}

public class DetailsTester {
	public static void main(String[] args) {
		MobileDetails details = new MobileDetails();
		details.setBrandname("Mi");
		details.setMobileid(10);
		details.setModelname("Note 11 pro");
		details.setPrice(20000.00);
		details.setMemory("128GB");
		details.setSerialno("S0G5QL");
		details.setNoofcamera(4);

		MobileDetails details1 = new MobileDetails();
		details1.setBrandname("Poco");
		details1.setMobileid(11);
		details1.setModelname("X3 pro");
		details1.setPrice(21000.00);
		details1.setMemory("64GB");
		details1.setSerialno("P0G5QL");
		details1.setNoofcamera(5);


		MobileDeatilsCRUD crud = new MobileDeatilsCRUD();

		crud.getAll();
		crud.deletebyMobileId(10);
		crud.updateBySerialNo("S0G5QL", "P8UT7W");

		crud.getAll();
	}

}
```