NSBM Green University

Faculty of Computing

<MIS>

Module Code & Name

Object Oriented Programming with Java

Module Lecturer: Mr Mohamed Shafraz

StudentName:Ushan Akalanka

Student ID: 22743

OOP with Java – Revision Questions

1)

```
//filename: Date.java
// Date class
public class Date {
private int month;
private int day;
private int year;
public Date(int myMonth,int myDay, int myYear) {

month = myMonth;
day = myDay;
year = myYear;
}

public void setMonthDate(int myMonth) {
month = myMonth;
}

public int getMonthDate() {
return month;
}

public void setDayDate(int myDay) {
day = myDay;
}

public int getDayDate() {
return month;
}

public void setYearDate(int myYear) {
year = myYear;
}

public int getYearDate() {
return year;
}

public void displayDate() {
System.out.printf(“%d/%d/%d”, month,day,year);
}
}
//filename: DateTest.java
// Date testing class with the main() method
import java.util.*;
public class DateTest {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
Date myDate = new Date(9, 11, 1998);

System.out.println(“Enter The Month”);
int myMonth = input.nextInt();
myDate.setMonthDate(myMonth);

System.out.println(“Enter the Date”);
int myDay = input.nextInt();
myDate.setDayDate(myDay);

System.out.println(“Enter the Year”);
int myYear = input.nextInt();
myDate.setYearDate(myYear);
myDate.displayDate();
}
}
```

2)

```
//filename: SavingAccount.java
// SavingAccount class
public class SavingsAccount {
public static double annualInterestRate;
private double savingsBalance;
public SavingsAccount() {
annualInterestRate = 0.0;
savingsBalance = 0.0;
}

public SavingsAccount(double intRate, double savBal) {
annualInterestRate = intRate;
```

```
savingsBalance = savBal;
}

public double calculateMonthlyInterest() {
double intRate = (savingsBalance * annualInterestRate/12);
savingsBalance = savingsBalance + intRate;
return intRate;
}

public static void modifyInterestRate(double newInteresRate) {
annualInterestRate = newInteresRate;
}

public void setSavingsBalance(double newBal) {
savingsBalance = newBal;
}
public double getSavingsBalance() {
return savingsBalance;
}
public double getAnnualInterestRate() {
return annualInterestRate;
}
}
//filename: SavingsAccountTest.java
// SavingsAccount testing class with the main() method
public class SavingsAccountTest {
public static void main(String[] args) {
SavingsAccount saver1 = new SavingsAccount();
SavingsAccount saver2 = new SavingsAccount();
saver1.setSavingsBalance(2000.00);
saver2.setSavingsBalance(3000.00);
SavingsAccount.modifyInterestRate(0.04);
saver1.calculateMonthlyInterest();
saver2.calculateMonthlyInterest();
System.out.printf("New Balance for
Saver1=%f\n",saver1.getSavingsBalance());
System.out.printf("New Balance for
Saver2=%f\n",saver2.getSavingsBalance());
```

```
SavingsAccount.modifyInterestRate(0.05);
saver1.calculateMonthlyInterest();
saver2.calculateMonthlyInterest();
System.out.printf("New Balance for
Saver1=%f\n",saver1.getSavingsBalance());
System.out.printf("New Balance for
Saver2=%f\n",saver2.getSavingsBalance());

}
}
```

3)

a)

```
//filename: Car.java
//Car class
public class Car {
private int speed;
private double regularPrice;
private String color;

public Car (int Speed,double regularPrice,String color) {
this.speed = Speed;
this.regularPrice = regularPrice;
this.color = color;
}

public double getSalePrice() {
return regularPrice;
}
}
```

b)

```
package objectOrientedExercises;


public class Truck extends Car {
```

```
int weight;


public Truck(int speed, double regularPrice, String colour, int weight)
{
        super(speed, regularPrice, colour);
        this.weight = weight;
}
public double getSalePrice() {
        if (weight > 2000) {
                double discountPrice = super.regularPrice * 0.10; // 10% of
regular


                return (super.regularPrice - discountPrice);


        } else {
                double discountPrice = super.regularPrice * 0.20; // 20% of
regular

        return (super.regularPrice - discountPrice);
        }
}


}
```

c)

```
//filename: Ford.java
// Ford class, subclass of Car
public class Ford extends Car {
private int year;
private int manufacturerDiscount;

public Ford (int Speed,double regularPrice,String color, int year, int
manufacturerDiscount) {
super (Speed,regularPrice,color);
this.year = year;
this.manufacturerDiscount = manufacturerDiscount;
}

public double getSalePrice() {
return (super.getSalePrice() – manufacturerDiscount);
}
}
```

d)

```
//filename: Sedan.java
// Sedan class, subclass of Car
public class Sedan extends Car {
private int length;

public Sedan (int Speed,double regularPrice,String color, int length) {
super (Speed,regularPrice,color);
this.length = length;
}

public double getSalePrice() {
if (length > 20) {
return super.getSalePrice() – (0.05 * super.getSalePrice());
}
else {
return super.getSalePrice() – (0.1 * super.getSalePrice());
}
}
}
```

e)

```
//filename: MyOwnAutoShop.java
// Testing class with the main() method
public class MyOwnAutoShop {
(int Speed,double regularPrice,String color, int year, int
manufacturerDiscount)
public static void main(String[] args) {
Sedan mySedan = new Sedan(160, 20000, “Red”, 10);
Ford myFord1 = new Ford (156,4452.0,”Black”,2005, 10);
Ford myFord2 = new Ford (155,5000.0,”Pink”,1998, 5);
Car myCar – new Car (555, 56856.0, “Red”);
System.out.printf(“MySedan Price %.2f”, mySedan.getSalePrice());
System.out.printf(“MyFord1 Price %.2f”, myFord1.getSalePrice());
System.out.printf(“MyFord2 Price %.2f”, myFord2.getSalePrice());
System.out.printf(“MyCar Price %.2f”, myCar.getSalePrice());
}
}
```

4)

4.1)

```
class Shape
{
void draw()
{
System.out.println(“Shape draw()”);
}
void erase()

{
System.out.println (“ Shape erase()”);
}
}
class Circle extends Shape
{
void draw()
{
System.out.println (“Circle draw()”);
}
void erase()
{
System.out.println (“Circle erase()”);
}
}
class Triangle extends Shape
{
void draw()
{
System.out.println(“Triangle erase()”);
}
}
class Square extends Shape
{
void draw()
{
System.out.println (“Square draw()”);
}
void erase()
{
System.out.println (“Square erase()”);
}
}
public class Shapes
{
public static Shape randshape()
{
switch((int)(Math.random()*3))
{
case 0: return new Circle();
case 1: return new Square();
case 2: return new Triangle();
default : System.out.println(“default”);
return new Shape();
}
}
public static void main (String arg[])
{
Shape s[] = new Shape[9];
for(int i = 0;i< s.length; i++) s[i] = randshape(); for(int i= 0;i < s.length;
i++) s[i].draw(); } }
```

4.2) abstract class Bike{

```
 abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

4.3)

```
abstract class debuggable{
abstract void dump()
{
System.out.println("debuggable error: no dump() defined for the
class");
}
}
class X extends debuggable{
private int a,b,c;
public:
X( int aa =0,int bb=0,int cc=0)
{
a = aa;
b = bb;
c = cc;
}
void dump()
{
Systen.out.println( "a = " + a +"b=" +b+ "c=" +c);
}
}
class Y extents debuggable{
private int i,j,k;
public:
Y( int ii =0,int jj=0,int kk=0)
{
i = ii;
j = jj;
k = kk;
}
void dump()
{
Systen.out.println( "i = " + i +"j=" +j+ "k=" +k);
}
}
class Z extents debuggable{
private int p,q,r;
public:
```

```
Y( int pp =0,int qq=0,int rr=0)
{
p = pp;
q = qq;
r = rr;
}
void dump()
{
Systen.out.println( “p = ” + p +”q=” +q+ “r=” +r);
}
}
class abstdemo
{
public static void main(String arg[])
{
X x(1,2,3);
Y y(2,4,5);
Z z;
x = new X;
y = new Y;
z = new Z;
x.dump();
y.dump();
z.dump();
}
```

5)

5.1)

```
// One interface an extend another.

interface A
{
void meth1();
void meth2();
}
// B now includes meth1() and meth2()–it adds meth3().
interface B extends A
{
void meth3();
}
// This class must implement all of A and B
class MyClass implements B
{
public void meth1 ( )
{
System.out.println(“Implement meth1().”);
}
public void meth2()
{
System.out.println (“Implement meth2().”);
}
public void meth3()
{
System.out.println (“Implement meth().” );
}
}
class IFExtend
{
public static void main(String arg[])
{
MyClass ob = new MyClass();
ob.meth1();
ob.meth2();
ob.meth3();
}
}
```

5.2)

```
interface AnimalEat {
  void eat();
}
interface AnimalTravel {
  void travel();
}
class Animal implements AnimalEat, AnimalTravel {
  public void eat() {
    System.out.println("Animal is eating");
  }
  public void travel() {
    System.out.println("Animal is travelling");
  }
}
public class Demo {
  public static void main(String args[]) {
    Animal a = new Animal();
    a.eat();
    a.travel();
  }
}
```

5.3)

```
// Interface
public interface Test
{
public int square(int a);
}
// Implements
class arithmetic implements Test
{
int s = 0;
public int square(int b)
{
System.out.println(“Inside arithmetic class – implemented method
square”);
System.out.println(“Square of “ + “ is “+s);
return s;
}
void armeth()
{

System.out.println(“Inside method of class Arithmetic”);
}
}

// use the object
class ToTestInt
```

```
{
public static void main(String a[])
{
System.out.println("calling from ToTestInt class main
method");
Test t = new arithmetic();
System.out.println("=================================");
System.out.println("created object of test interface – reference
Arithmetic class ");
System.out.println("Hence Arithmetic class method square
called");
System.out.println("This object cannot call armeth method of
Arithmetic class");
System.out.println("====================================");
t.square(10);
System.out.println("====================================");
}
}
```

5.4)

```
class Outer{
String so = ("This is Outer Class");
void display()
{
System.out.println(so);
}
void test(){
Inner inner = new Inner();
inner.display();
}
//this is an inner class
class Inner{
String si =("This is inner Class");
void display(){
System.out.println(si);
}
}
}
class InnerClassDemo{
public static void main(String args[]){
Outer outer = new Outer();
outer.display();
outer.test();
}

}
```

6)

6.1)

```
class NegTest
{
public static void main(String a[])
{
try
{
int a1[] = new int[-2];
System.out.println("first element : "+a1[0]);
}
catch(NegativeArraySizeException n)
{
System.out.println(" generated exception : " + n);
}
System.out.println(" After the try block");
}
}
```

6.2)

```
public class MultipleCatchBlock2 {

    public static void main(String[] args) {

      try{
         int a[]=new int[5];

       System.out.println(a[10]);
           }
        catch(ArithmeticException e)
           {
        System.out.println("Arithmetic Exception occurs");
            }
        catch(ArrayIndexOutOfBoundsException e)
            {
        System.out.println("ArrayIndexOutOfBounds Exception occurs");
             }
         catch(Exception e)
            {
        System.out.println("Parent Exception occurs");
            }
        System.out.println("rest of the code");
    }
}
```

6.3)

```
import java.io.*;
class Parent{
  void msg()throws ArithmeticException {
    System.out.println("parent method");
  }
}

public class TestExceptionChild2 extends Parent{
  void msg()throws Exception {
    System.out.println("child method");
  }

  public static void main(String args[]) {
   Parent p = new TestExceptionChild2();

   try {
   p.msg();
   }
   catch (Exception e){}

  }
}
```

6.4)

```
public class TestFinallyBlock1{
      public static void main(String args[]){


      try {


    System.out.println("Inside the try block");


      int data=25/0;
       System.out.println(data);
      }
      catch(NullPointerException e){
        System.out.println(e);
     }


      finally {
        System.out.println("finally block is always executed");
      }


      System.out.println("rest of the code...");
      }
    }
```

6.5)

```
public void myMethod()
{
  try {
    // Statements that might throw an exception
  }
  catch (ArithmeticException e) {
    // Exception handling statements
  }
  catch (NullPointerException e) {
    // Exception handling statements
  }
}
```

6.6)

```
// class representing custom exception
class MyCustomException extends Exception
{

}

// class that uses custom exception MyCustomException
public class TestCustomException2
```

```
{
    // main method
    public static void main(String args[])
    {
        try
        {
            // throw an object of user defined exception
            throw new MyCustomException();
        }
        catch (MyCustomException ex)
        {
            System.out.println("Caught the exception");
            System.out.println(ex.getMessage());
        }

        System.out.println("rest of the code...");
    }
}
```