Assignment-6

Q1. Create a user-defined package named 'geoshapes'. In this package, define the following classes with necessary methods such as area and perimeter.

i) Circle ii) Rectangle iii) Triangle iv) Sphere import the package and

demonstrate the use of objects of each defined class.

-> Circle package com.example.geoshapes; import

java.util.*; public class circle

{

double x,y,radius;

Scanner sc = new Scanner(System.*in*); public void getR()

{

System.*out*.print("Enter x and y cordinate of Circle : "); x = sc.nextDouble(); y =

sc.nextDouble(); System.*out*.print("Enter Radius : "); radius = sc.nextDouble();

}

public double area()

{

return 3.14*radius*radius;

}

public double perimeter()

{return 2*3.14*radius;

}}

-> Rectangle package

com.example.geoshapes;

import java.util.*; public

class Rectangle

{

double length,width;

Scanner sc = new Scanner(System.in); public

void getLW()

{

System.out.print("Enter Length : "); length

= sc.nextDouble();

System.out.print("Enter Width : "); width

= sc.nextDouble();

}

```
public double area()
{
return length*width;
}
public double perimeter()
{
return 2*(length+width);
}
}
```

-> Triangle package com.example.geoshapes; import java.util.*; public class Triangle {

```
double base,height; Scanner sc = new Scanner(System.in); public void getBH()
{
System.out.print("Enter Base : "); base = sc.nextDouble();
System.out.print("Enter Height : ");
height = sc.nextDouble();
}
public double area()
{
return 0.5*base*height;
}
public double perimeter()
{
return base+height+Math.sqrt(base*base+height*height);
}
}
```

iv) Sphere package com.example.geoshapes; import java.util.*; public class Sphere { double radius;

```
Scanner sc = new Scanner(System.in); public
void getBHS()
{
System.out.print("Enter radius : "); radius
= sc.nextDouble();
}
public double area()
{
return 4*3.14*(radius*radius);
}
}
MAIN() :- package
com.example.geoshapes;
import java.util.*; public class
prog1 { public static void
main(String args[])
{
Scanner sc = new Scanner(System.in); circle
C = new circle();
Rectangle R = new Rectangle();
Triangle T = new Triangle();
Sphere P = new Sphere();
int choice; while(true)
{
System.out.println("1. Find area & Perimeter of Circle");
System.out.println("2. Find area & Perimeter of Rectangle");
System.out.println("3. Find area & Perimeter of Triangle");
System.out.println("4. Find area of Sphere");
System.out.println("5. Exit");
System.out.print("Enter Your Choice : ");
choice = sc.nextInt(); switch(choice)
{
case 1:
System.out.print("\n");
C.getR();
```

```
System.out.print("Area of Circle : "+C.area());
System.out.println("\n");
System.out.print("Perimeter of Circle : "+C.perimeter());
System.out.println("\n"); break; case 2:
System.out.print("\n");
R.getLW();
System.out.print("Area of Rectangle : "+R.area());
System.out.println("\n");
System.out.print("Perimeter of Rectangle : "+R.perimeter());
System.out.println("\n"); break; case 3:
System.out.print("\n");
T.getBH();
System.out.print("Area of Triangle : "+T.area());
System.out.println("\n");
System.out.print("Perimeter of Triangle : "+T.perimeter());
System.out.println("\n"); break; case 4:
System.out.print("\n");
P.getBHS();
System.out.print("Area of Sphere: "+P.area());
System.out.println("\n"); break; case 5 :
System.exit(0); break; default:
System.out.println("Wrong Choice");
}
}
}
}
```

Q2. Create a user-defined package named as 'coordinates'. Define two Classes namely Cartesian and Polar in coordinate package with all necessary methods. Define methods toPolar() and toCartesian() in Cartesian and Polar class respectively for converting in to respective types of coordinate. Demonstrate the conversion in both way by importing the package and using method of these objects.

-> package com.example.coordinates;

import java.util.*; public

class Polor {

```
Scanner sc = new Scanner(System.in);
double PI = 3.141592; double x,y;
public void toCartesian(double r, double theta)
{
theta = theta * (PI / 180.0);
x = r * Math.cos(theta); y
= r * Math.sin(theta);
System.out.println("Cartesian Co-ordinates : ( " + x + ", " + y + " )");
}
}
package com.example.coordinates;
import java.util.*; public class
Cartesian {
Scanner sc = new Scanner(System.in);
double PI = 3.141592; double r,theta;
public void toPolor(double x, double y)
{
double r = Math.sqrt(x*x + y*y);
double theta = Math.atan2(y, x); theta
= theta * (PI / 180.0);
System.out.println("Polor Co-ordinates : ( " + r + ", " + theta + " )");
} }
//MAIN CLASS
package com.example.coordinates;
import java.util.*;

public class prog2 { public static
void main(String args[])
{
Scanner sc = new Scanner(System.in);
Cartesian C= new Cartesian();
Polor P = new Polor(); int
choice; while(true)
{
System.out.println("1. topolor");
System.out.println("2. toCartesian");
```

```
System.out.println("3. Exit");
System.out.print("Enter Your Choice :
"); choice = sc.nextInt(); switch(choice) {
case 1:
System.out.print("\n");
System.out.print("Enter Cartesian coordinates(X, Y) : ");
double x = sc.nextDouble(); double y = sc.nextDouble();
System.out.println("Cartesian to Polor form");
C.toPolor(x,y);
System.out.println("\n");
break; case 2:
System.out.print("\n");
System.out.print("Enter Polar Coordinates(r, theta) : ");
double r = sc.nextDouble(); double theta =
sc.nextDouble();
System.out.println("Polor to Cartesian form");
P.toCartesian(r,theta);
System.out.println("\n");
break; case 3:
System.exit(0); break;
default:

System.out.println("Wrong Choice");
} }
}
}
```

Q3. Create an interface with name 'LinearDS' with the following member and methods: i) MAXSIZE – indicate the maximum capacity of the data structure ii) add(element) - the method to add an element to the datastructure (as per its rule) iii) remove() - the method to remove an element from the data structure (as per its rule) iv) displayElement() - the method to display all the element of the data structure. Implement your interface to define two classes MyStack and MyQueue to implement all operations of a stack and queue using LinearDS respectively.

->

```
package com.example.prog3;
import java.util.*; interface
LinerDS { int MAXSIZE = 5; void
add(int element); int remove();
void displayElement();
```

```
}
class MyStack implements LinerDS {
private int stack[]; private int top;
MyStack(){ stack = new
int[MAXSIZE]; top = -1;
}
public void add(int element) { if(top
== MAXSIZE-1) {
System.out.println("Stack Overflows");
}
else stack[++top] =
element;
}
public int remove() { if(top
== -1) {
System.out.println("Stack Underflow"); return
-1;
}
else return
stack[top--];
}
public void displayElement() { if(top
== -1)
System.out.println("Stack is empty"); else
{
System.out.println("\nStack element is ");
for (int i = top; i >= 0; i--)
System.out.println(" " + stack[i] + " ");
System.out.println();
}
}
}
class MyQueue implements LinerDS {
private int que[]; private int front;
private int rear; MyQueue() { que =
```

```
new int[MAXSIZE]; front = -1; rear = -
1;
}
public void add(int element) { if(rear
== MAXSIZE-1)
System.out.println("Queue Overflow");
else { front = 0; rear
= rear+1; que[rear]
= element;
}
}
public int remove() { int item;
if (front == -1 || front > rear)
{
System.out.println("Queue Underflow"); return -1;
} else { item =
que[front];
if(front == rear) {
front = -1; rear = -
1 ;
}
else front =
front + 1;
}
return item;
}
public void displayElement() {
int i; if(rear
== -1)
System.out.println("Empty Queue");
else {
System.out.print("\nQueue element is : " ); for(i
= front; i <= rear; i++)
System.out.print(" " +que[i]);
System.out.println();
```

```
}
}
}
public class StackNQueue { public
static void main(String args[]) { int
item;
Scanner sc = new Scanner(System.in);
MyStack MS = new MyStack(); MyQueue
MQ = new MyQueue();
int choice; do
{
System.out.println("1. Add Element in Stack");
System.out.println("2. Remove Element from Stack");
System.out.println("3. Display Stack Element"); System.out.println("4. Add
Element in Queue");
System.out.println("5. Remove Element from Queue");
System.out.println("6. Display Queue Element");
System.out.println("7. Exit");
System.out.print("Enter Your Choice : ");
choice = sc.nextInt(); switch (choice) {
case 1:
System.out.print("\nEnter element in Stack : ");
item = sc.nextInt(); MS.add(item); break; case
2: item = MS.remove();
if (item != -1)
System.out.println("\nRemoved element from Stack is : " + item);
break; case 3: MS.displayElement(); break; case 4:
System.out.print("\nEnter element in Queue : ");
item = sc.nextInt(); MQ.add(item); break; case 5:
item = MQ.remove();
if (item != -1)
System.out.println("\nRemoved element from Queue is : " + item);
break; case 6: MQ.displayElement(); break; case 7:
System.out.println("Exiting..."); break;
}
```

```
}while (choice!=7);
}
}
```

Assignment-7

Q1. Demonstrate the printing of alphabets from ‘a’ to ‘z’ using multithreading environment.

->

```
class printAtoZ implements Runnable{
Thread t; printAtoZ(){
t=new Thread(this,"print A to Z");
t.start(); }
public void run(){ try{
for(char i='a';i<='z';i++)
System.out.println(i);
Thread.sleep(1000);
}
catch (InterruptedException e)
{
e.printStackTrace();
} }
}
class Assign7_1{
public static void main(String[] args)
{
```

```
new printAtoZ();
}
}
```

Q2. Demonstrate the Shortest Job First Process Scheduling Algorithm using java multithreading. Here, assume that job size is decided by the numbers elements (n) a method is going to print from 1 to n. For example, a method called to print elements from 1 to 5 has higher priority than a method called to print from 1 to 10.

->

```
class PrintClass implements Runnable {
Thread t; int m;
String p;
PrintClass(int priority, int m) { t = new
Thread(this, "Print Thread " + m);
t.setPriority(priority);
t.start();
this.m= m;
}
public void run() { for
(int i = 1; i <= m; i++)
System.out.println(t.getName() + " " + i);
} }
public class Assign_7_2{
public static void main(String[] args) {
new PrintClass(10, 100); new
PrintClass(20, 50);
}
}
```

Q3. Demonstrate the use of synchronized block for a Inventory class where one and only one thread can update the value of a stock item at a any instance of time.

```
class Inventory {
int stock;
Inventory() { stock
= 0;
}
synchronized void addFromStock() {
if (stock > 20) {
```

```
System.out.println("Overflow");
}
System.out.println("Added to stock, Updated stock:" + (stock++));
}
synchronized void removeFromStock() { if
(stock < 0) {
System.out.println("Underflow");
}
System.out.println("Remove to stock, Updated stock: " + (--stock));
}
}
class Add implements Runnable {
Thread t; int n;
String s;
Inventory invntry;
Add(int n, Inventory invntry) { t =
new Thread(this, "Add thread");
this.n = n; this.invntry = invntry;
t.start(); } public void
run() { for (int i = 0; i <
n; i++) {
invntry.addFromStock();
} }
} class Remove implements Runnable
{ Thread t; int n;
String s;
Inventory invntry; Remove(int n,
Inventory invntry) { t = new
Thread(this, "Remove thread"); this.n =
n; this.invntry = invntry;
t.start(); } public void run()
{ for (int i = 0; i < n; i++) {
invntry.removeFromStock()
;
} } } class Assign_7_3 { public static
void main(String[] args) { Inventory
invntry = new Inventory(); new
```

```
Add(20, invntry); new Remove(20,
invntry);
} }
```

Output:

```
Added to stock, Updated stock:0
Added to stock, Updated stock:1
Added to stock, Updated stock:2
Added to stock, Updated stock:3
Added to stock, Updated stock:4
Added to stock, Updated stock:5
Added to stock, Updated stock:6
Added to stock, Updated stock:7
Added to stock, Updated stock:8
Added to stock, Updated stock:9
Added to stock, Updated stock:10
Added to stock, Updated stock:11
Added to stock, Updated stock:12
Added to stock, Updated stock:13
Added to stock, Updated stock:14
Added to stock, Updated stock:15
Added to stock, Updated stock:16
Added to stock, Updated stock:17
Added to stock, Updated stock:18
Added to stock, Updated stock:19
Remove to stock, Updated stock: 19
Remove to stock, Updated stock: 18
Remove to stock, Updated stock: 17
Remove to stock, Updated stock: 16
Remove to stock, Updated stock: 15
Remove to stock, Updated stock: 14
Remove to stock, Updated stock: 13
Remove to stock, Updated stock: 12
Remove to stock, Updated stock: 11
Remove to stock, Updated stock: 10
Remove to stock, Updated stock: 9
Remove to stock, Updated stock: 8
Remove to stock, Updated stock: 7
```

Remove to stock, Updated stock: 6

Remove to stock, Updated stock: 5

Remove to stock, Updated stock: 4

Remove to stock, Updated stock: 3

Remove to stock, Updated stock: 2

Remove to stock, Updated stock: 1

Remove to stock, Updated stock: 0