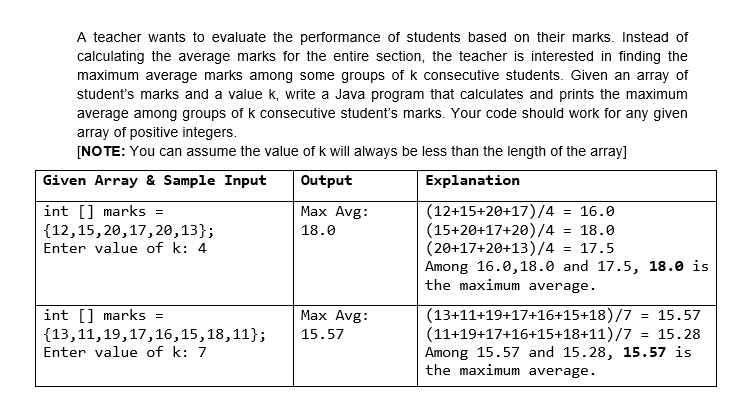
# Linear and Multidimensional array

### **Question 1:**



### **Question 2:**

Construct a 2D array by taking the row, column and array elements as double inputs from the user. Print the 2D array. Find the maximum number at each row and print the maximum numbers.

**Hint**: You may use Arrays.deepToString() to print the 2D array.

Ex: System.out.println(Arrays.deepToString(array\_name));

| Sample Input | Sample Output |
| --- | --- |
| row = 3  col = 3  Enter the double values:  -1.3  2.0  3.3  4.1  5.3  5.4  7.1  8.2  -2.0 | 2D array:  [[-1.3, 2.0, 3.3],  [4.1, 5.3, 5.4],  [7.1, 8.2, -2.0]]  Maximum number at row 0 : 3.3  Maximum number at row 1 : 5.4  Maximum number at row 2 : 8.2 |

### **Question 3:**

Execute the matrix multiplication of two given 2D arrays, store the result in another array and print the elements from the resulting array.

Your code should work for any given arrays.

| Given Arrays | Output |
| --- | --- |
| arr1 = {{1, 2, 3},  {4, 5, 6},  {3, 1, 3}}  arr2 = {{5, 1},  {1, 2},  {4, 0}} | 19 5  49 14  28 5 |

# OOP Before Mid

**Reverse Engineering Tasks**

### **Question 1:**

Write the design class.

1. Assume a whatsapp group can have exactly 5 members.
2. The check() method checks if a person exists in that Whatsapp group.

| Tester Code | Output |
| --- | --- |
| WhatsappGroup w1 = new WhatsappGroup();  System.*out*.println("1=================");  w1.add\_members("Bob");  w1.add\_members("Alice");  w1.add\_members("Carol");  w1.add\_members("Tony");  w1.add\_members("Riley");  w1.add\_members("Rob");  System.*out*.println("2=================");  System.*out*.println(w1.display());  System.*out*.println("3=================");  System.*out*.println(w1.check("Bob"));  System.*out*.println("4=================");  w1.send\_message("Bob", "Hello!");  System.*out*.println("5=================");  w1.send\_message("Rob", "Can I send a message?"); | 1========================  A member has been successfully added  A member has been successfully added  A member has been successfully added  A member has been successfully added  A member has been successfully added  Cannot be added. Sorry  2========================  Bob  Alice  Carol  Tony  Riley  3========================  true  4========================  Bob can send a message. He wants to say Hello!  5========================  Rob cannot send a message. |

**Instance vars - array, index**

**display() - return a string of members from the array**

### **Question 2:**

Design the **Game** class such with necessary properties so that the given output is produced

Note:

1. **assignDetails()** assigns the Game name, duration of the game (in minutes) and number of segments of that game.

2. **setSegment()** sets a segment at a time for the game

| Tester Code | Output |
| --- | --- |
| public class GameTester  {  public static void main(String [] args){  Game game1 = new Game();  System.out.println("1============");  game1.assignDetails("Soccer Pro", 90.0, 3);  System.out.println("2============");  game1.setSegment("Warm Up");  game1.setSegment("Intermediary");  game1.setSegment("Hard");  game1.setSegment("Very Hard");  System.out.println("3============");  game1.gameInfo();  System.out.println("4============");  Game game2 = new Game();  game2.assignDetails("Go Chess", 60.0, 2);  game2.setSegment("Phase 1");  game2.setSegment("Phase 2");  System.out.println("5============");  game2.gameInfo();  System.out.println("6============");  System.out.println(game2.endGame());  System.out.println("7============");  game2.gameInfo();  }  } | 1============  2============  Cannot add the segment: Very Hard  3============  Game: Soccer Pro  Time left: 90.0 mins  Segments selected:  Warm Up  Intermediary  Hard  4============  5============  Game: Go Chess  Time left: 60.0 mins  Segments selected:  Phase 1  Phase 2  6============  Go Chess game has ended. No time left.  7============  Game: Go Chess  Time left: 0.0 mins |

### **Question 3:**

Write the design classes following the given tester code and output. Assume that a user can receive maximum 7 connection requests and accept maximum 4 connection requests.

| Tester Code | Output |
| --- | --- |
| public class LinkedInTester {  public static void main(String[] args) {  LinkedInUser l1 = new LinkedInUser("Iron man");  LinkedInUser l2 = new LinkedInUser("Spider man");  LinkedInUser l3 = new LinkedInUser("Deadpool");  LinkedInUser l4 = new LinkedInUser("Wolverine");  LinkedInUser l5 = new LinkedInUser("Thor");  LinkedInUser l6 = new LinkedInUser("Ant man");  LinkedInUser l7 = new LinkedInUser("Captain America");  LinkedInUser l8 = new LinkedInUser("Ultron");  LinkedInUser l9 = new LinkedInUser("Loki");  LinkedInUser l10 = new LinkedInUser("Dr. Strange");  System.*out*.println("1====================");  LinkedInUser userList [] = {l2, l3, l4, l5, l6, l7, l8, l9, l10};  l1.receiveConnectionRequest(userList);  System.*out*.println("2====================");  l1.acceptConnectionRequest(l3);  System.*out*.println("3====================");  l1.acceptConnectionRequest(l9);  System.*out*.println("4====================");  l1.acceptConnectionRequest(l4, l10);  System.*out*.println("5====================");  l1.acceptConnectionRequest(l8, l2);  System.*out*.println("6====================");  l1.acceptConnectionRequest(l5);  System.*out*.println("7====================");  System.*out*.println(l1.connectionDetails());  }  } | A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  A LinkedIn user has been created!  1====================  Iron man received a connection request from Spider man.  Iron man received a connection request from Deadpool.  Iron man received a connection request from Wolverine.  Iron man received a connection request from Thor.  Iron man received a connection request from Ant man.  Iron man received a connection request from Captain America.  Iron man received a connection request from Ultron.  A user can receive maximum 7 connection requests.  2====================  Iron man accepted the connection request from Deadpool.  3====================  Iron man did not receive a connection request from Loki.  4====================  Iron man accepted the connection request from Wolverine.  Iron man did not receive a connection request from Dr. Strange.  5====================  Iron man accepted the connection request from Ultron.  Iron man accepted the connection request from Spider man.  6====================  A user can accept maximum 4 connection requests.  7====================  Iron man connected with the following:  Deadpool, Wolverine, Ultron, Spider man |

**Tracing Task**

### **Question 4:**

Design Class:

public class oop\_beforeMid\_tracing {

public int temp = 4;

public int sum;

public int y;

public int x;

public void methodA(int y){

int [] n = {2,5};

int x = 1;

this.y = y + this.methodB(x, this.y)+(temp++) + this.y; //what if just y?

x = this.x + 2 + (++n[0]);

sum = sum + x + y;

System.*out*.println( x + " " + y+ " " + sum);

}

public int methodB(int m, int n){

this.y = this.y + m;

this.x = this.y + 2 + temp - n;

System.*out*.println(this.x + " " +sum+ " " +y);

return this.y;

}

}

TesterClass:

public class Tester

{

public static void main(String[] args) {

oop\_beforeMid\_tracing t1 = new oop\_beforeMid\_tracing();

t1.methodA(5);

oop\_beforeMid\_tracing t2 = new oop\_beforeMid\_tracing();

t2.methodA(3);

oop\_beforeMid\_tracing t3 = null;

t3 = t1;

t3.methodB(2, 4);

}

}

### **Question 5:**

**Design Class:**

public class TracingProblem

{

public int temp = 4;

public int y = 2, x, sum;

public TracingProblem()

{

this.x = 3;

sum += x;

}

public TracingProblem(TracingProblem obj)

{

this.y = temp - 2;

obj.y = x + temp - 1;

obj.methodA(this.y); //what if this.methodA(this.y) or methodA(this.y)?

}

public void methodA(int y){

int [] n = {2,5};

int x = 1;

this.y = y + this.methodB(n)+(temp++) + this.y; //what if just y?

x = this.x + (++n[0]);

sum = sum + x + y;

System.out.println( x + " " + y+ " " + sum);

}

public void methodB(int m, int n){

TracingProblem t4 = new TracingProblem(this);

this.y = this.y + m;

this.x = t4.temp + temp - n;

System.out.println(this.x + " " +sum+ " " +y);

}

public int methodB(int [] arr){

this.y += arr[1];

arr[0] ++;

this.x = this.y + 1 + temp;

System.out.println(this.x + " " +sum+ " " +y);

return this.y --;

}

}

**TesterClass:**

public class Tester

{

public static void main(String[] args) {

TracingProblem t1 = new TracingProblem();

TracingProblem t2 = new TracingProblem(t1);

TracingProblem t3 = null;

t3 = t2;

t3.methodB(2, 4);

t2.methodA(3);

}

}

[**Solution**](https://docs.google.com/spreadsheets/d/1pxUOB7yX5ufV2llKQeNHY0dZ4rgMzEoXfleULI06Mr0/edit?usp=sharing)

### **Question 6:**

**Design Class:**

public class TracingProblem

{

public int temp = 4, x, sum = 2;

public TracingProblem()

{

this.x = 3;

sum += x;

temp --;

}

public void methodA(TracingProblem t5){

int x = 1;

methodB(this);

this.temp = t5.temp + x;

this.x = x + (++sum);

sum = sum + this.x;

System.out.println( this.x + " " + temp+ " " + sum);

}

public void methodB(TracingProblem t1){

this.temp = t1.temp + t1.x;

this.x = t1.sum + sum;

t1.x += this.x;

System.out.println(this.x + " " +this.temp+ " " +t1.x);

}

}

**Tester Class:**

public class Tester

{

public static void main(String[] args) {

TracingProblem t1 = new TracingProblem();

TracingProblem t3 = new TracingProblem();

t3.methodA(t1);

}

}

# OOP After Mid

### **Question 7:**

[Solution](https://docs.google.com/spreadsheets/d/1wXsebfku-bMbOxWEv9ZVnHtRd0p-bHADJuuCIkXVfrk/edit?gid=0#gid=0)

| Parent class:  class A {  public static int temp = 4;  public static int x = -10;  public int sum, y ;  public A() {  y = temp - 2;  sum = temp + 1;  this.temp -= 2;  }  public A(int a, int b) {  y = temp - a;  sum = temp + b;  }  public void methodA(int m, int n) {  int x = 0;  y = y + m + (temp++);  x = 1 + n;  sum = sum + this.x + y;  methodB(x, y);  System.out.println(x + " " + y + " " + sum);  }  public void methodB(int m, int n) {  int y = 0;  y = m + this.y;  x = n + (++temp);  sum += temp;  System.out.println(x + " " + y + " " + sum);  }  } | Child Class:  class B extends A {  public static int x = 0;  public int sum = -6;  public B() {  sum = 0;  y = B.temp + 3;  super.sum = 3 + x + 2;  temp -= 2;  }  public B(B b) {  super(b.sum, 4);  y = temp + this.x;  sum += super.sum;  }  @Override  public void methodB(int a, int b) {  x += a;  y += b;  sum = x + y + super.x;  System.out.println(x + " " + y + " " + sum);  }  public void methodC(int m, int n) {  int x = 1;  this.x += x;  y = n + (++ A.temp);  super.methodB(y, n - 1); //what if methodB()?  sum = x + y + sum;  System.out.println(this.x + " " + y + " " + sum);  }  } |
| --- | --- |
| Tester Code:  public class Tester {  public static void main(String[] args) {  A a1 = new A();  a1.methodA(1, 3);  B b1 = new B();  b1.methodA(2, 4);  b1.methodC(5, -1);  B b2 = new B(b1);  b2.methodB(3, 8);  }  } | |

Polymorphism Tracing

**Casting rules**:

### 

### **Question 8:**

| Design Classes | Tester Code |
| --- | --- |
| class Gadget {  String name = "Just a gadget";  public void m1()  {  System.out.println("Method1 of Gadget");  }  public void m2()  {  System.out.println(name);  m3();  }  public void m3()  {  System.out.println("Method3 of Gadget");  }  public String toString()  {  return "Name is: "+name;  }  }  class Computer extends Gadget{  String name = "Computer";  public void m3()  {  System.out.println(super.name);  m1();  }  public String toString()  {  return "Name is: "+name;  }  public void m4(int i)  {  System.out.println(i \* 3);  }  }  public class PersonalComputer extends Computer{  String name = "Personal Computer";  public void m1()  {  System.*out*.println(name);  System.*out*.println(super.toString());  System.*out*.println(this);  }  public String toString()  {  return "Name is: "+name;  }  }  class WorkComputer extends Computer{  public void m4(double i)  {  System.out.println(i \* 2);  System.out.println("Method4 of WorkComputer");  }  }  class Mobile extends Gadget{  String name = "Just a mobile";  public void m3()  {  System.out.println(name);  super.m3();  }  } | Object m1 = new Mobile();  Computer pc1 = new PersonalComputer();  Gadget c1 = new Computer();  Computer wc1 = new WorkComputer();  Gadget g1 = new Gadget();  System.out.println(pc1.name);  System.out.println(m1.name);  wc1.m4(10);  m1.m2();  c1.m2();  System.out.println(m1);  Object obj = (Object) c1;  System.out.println(obj.name);  System.out.println(obj.toString());  System.out.println(((Gadget) pc1).name);  System.out.println(((WorkComputer) wc1).name);  ((Gadget) pc1).m1();  ((Gadget) m1).m2();  ((Computer) m1).m2();  ((Object) g1).m2(); |

### **Question 9:**

Credits: A Former Faculty Member, CSE department

**Polymorphism is Fun**

**Assume that the following classes have been defined:**

| **public class Quadrilateral** |
| --- |
| **{** |
| **public String name = "Quadrilateral";** |
|  |
| **public void method1()** |
| **{** |
| **System.out.println("Quadrilateral 1");** |
| **}** |
| **public void method2()** |
| **{** |
| **System.out.println("Quadrilateral 2");** |
| **}** |
| **public void method3()** |
| **{** |
| **System.out.println("Quadrilateral 3");** |
| **}** |
| **public void method4()** |
| **{** |
| **System.out.println("Quadrilateral 4");** |
| **}** |
| **public String toString(){** |
| **this.method4();** |
| **return "This is Quadrilateral Class";** |
| **}** |
| **}** |

| **public class Trapezium extends Quadrilateral** |
| --- |
| **{** |
| **public String name = "Trapezium";** |
|  |
| **public void method1(){** |
| **System.out.println("Trapezium 1");** |
| **}** |
|  |
| **public String toString(){** |
| **return "This is a "+ name;** |
| **}** |
| **}** |

| **public class Kite extends Quadrilateral** |
| --- |
| **{** |
| **public String name = "Kite";** |
|  |
| **public void method1()** |
| **{** |
| **System.out.println("Kite 1");** |
| **}** |
| **public void method3()** |
| **{** |
| **System.out.println("Kite 3");** |
| **}** |
| **public void method4()** |
| **{** |
| **System.out.println("Kite 4");** |
| **}** |
| **}** |

| **public class Parallelogram extends Quadrilateral** |
| --- |
| **{** |
| **public String name = "Parallelogram";** |
|  |
| **public void method3()** |
| **{** |
| **System.out.println("Parallelogram 3");** |
| **super.method2();** |
| **method4();** |
| **}** |
| **}** |

| **public class Rhombus extends Parallelogram** |
| --- |
| **{** |
| **public String name = "Rhombus";** |
|  |
| **public void method1(){** |
| **System.out.println(this);** |
| **System.out.println("Rhombus 1");** |
| **}** |
|  |
| **public void method3(){** |
| **super.method2();** |
| **System.out.println("Rhombus 3");** |
| **}** |
|  |
| **}** |

| **public class Rectangle extends Parallelogram** |
| --- |
| **{** |
| **public String name = "Rectangle";** |
|  |
| **public void method2(){** |
| **method4();** |
| **System.out.println("Rectangle 2");** |
| **System.out.println(this);** |
| **}** |
|  |
| **public int compareTo(Rectangle a){** |
| **if(a instanceof Rectangle){** |
| **return 1;** |
| **}else{** |
| **return 0;** |
| **}** |
| **}** |
|  |
| **}** |

| **public class Square extends Rectangle** |
| --- |
| **{** |
| **public String name = "Square";** |
|  |
| **public void method1(){** |
| **method3();** |
| **System.out.println("Square 1");** |
| **}** |
|  |
| **public void method2(){** |
| **super.method2();** |
| **System.out.println("Square 2");** |
| **method3();** |
| **}** |
| **}** |

**And assume that the following variables have been defined:**

**Quadrilateral shape1 = new Quadrilateral();**

**Object shape2 = new Kite();**

**Quadrilateral shape3 = new Trapezium();**

**Object shape4 = new Parallelogram();**

**Parallelogram shape5 = new Rhombus();**

**Quadrilateral shape6 = new Rectangle();**

**Parallelogram shape7 = new Square();**

**Rectangle shape8 = new Square();**

**In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" which indicates three lines of output with "a" followed by "b" followed by "c". If the statement causes an error, fill in the right-hand column with either “CT” for “compile time error" or RE for "runtime error" to indicate when the error would be detected.**

| **Statement** | **Output** |
| --- | --- |
| **System.out.println(shape1.name);** |  |
| **System.out.println(shape2.name);** |  |
| **System.out.println(shape3.name);** |  |
| **System.out.println(shape4.name);** |  |
| **System.out.println(shape5.name);** |  |
| **System.out.println(shape6.name);** |  |
| **System.out.println(shape7.name);** |  |
| **shape1.method1();** |  |
| **shape1.method2();** |  |
| **shape1.method3();** |  |
| **shape1.method4();** |  |
| **shape2.method1();** |  |
| **shape2.method2();** |  |
| **shape2.method3();** |  |
| **shape2.method4();** |  |
| **shape3.method1();** |  |
| **shape3.method2();** |  |
| **shape3.method3();** |  |
| **shape3.method4();** |  |
| **shape4.method1();** |  |
| **shape4.method2();** |  |
| **shape4.method3();** |  |
| **shape4.method4();** |  |
| **shape5.method1();** |  |
| **shape5.method2();** |  |
| **shape5.method3();** |  |
| **shape5.method4();** |  |
| **shape6.method1();** |  |
| **shape6.method2();** |  |
| **shape6.method3();** |  |
| **shape6.method4();** |  |
| **shape7.method1();** |  |
| **shape7.method2();** |  |
| **shape7.method3();** |  |
| **shape7.method4();** |  |
| **System.out.println(shape8.compareTo(shape8));** |  |
| **((Quadrilateral) shape1).method1();** |  |
| **((Quadrilateral) shape2).method1();** |  |
| **((Quadrilateral) shape3).method1();** |  |
| **((Quadrilateral) shape4).method1();** |  |
| **((Quadrilateral) shape5).method1();** |  |
| **((Quadrilateral) shape6).method1();** |  |
| **((Quadrilateral) shape7).method1();** |  |
| **((Quadrilateral) shape1).method2();** |  |
| **((Quadrilateral) shape2).method2();** |  |
| **((Quadrilateral) shape3).method2();** |  |
| **((Quadrilateral) shape4).method2();** |  |
| **((Quadrilateral) shape5).method2();** |  |
| **((Quadrilateral) shape6).method2();** |  |
| **((Quadrilateral) shape7).method2();** |  |
| **((Quadrilateral) shape1).method3();** |  |
| **((Quadrilateral) shape2).method3();** |  |
| **((Quadrilateral) shape3).method3();** |  |
| **((Quadrilateral) shape4).method3();** |  |
| **((Quadrilateral) shape5).method3();** |  |
| **((Quadrilateral) shape6).method3();** |  |
| **((Quadrilateral) shape7).method3();** |  |
| **((Object) shape1).method1();** |  |
| **((Object) shape2).method1();** |  |
| **((Object) shape3).method1();** |  |
| **((Object) shape4).method1();** |  |
| **((Object) shape5).method1();** |  |
| **((Object) shape6).method1();** |  |
| **((Object) shape7).method1();** |  |
| **((Object) shape1).method2();** |  |
| **((Object) shape2).method2();** |  |
| **((Object) shape3).method2();** |  |
| **((Object) shape4).method2();** |  |
| **((Object) shape5).method2();** |  |
| **((Object) shape6).method2();** |  |
| **((Object) shape7).method2();** |  |
| **((Object) shape1).method3();** |  |
| **((Object) shape2).method3();** |  |
| **((Object) shape3).method3();** |  |
| **((Object) shape4).method3();** |  |
| **((Object) shape5).method3();** |  |
| **((Object) shape6).method3();** |  |
| **((Object) shape7).method3();** |  |
| **((Kite) shape1).method1();** |  |
| **((Kite) shape2).method1();** |  |
| **((Kite) shape3).method1();** |  |
| **((Kite) shape4).method1();** |  |
| **((Kite) shape5).method1();** |  |
| **((Kite) shape6).method1();** |  |
| **((Kite) shape7).method1();** |  |
| **((Kite) shape1).method2();** |  |
| **((Kite) shape2).method2();** |  |
| **((Kite) shape3).method2();** |  |
| **((Kite) shape4).method2();** |  |
| **((Kite) shape5).method2();** |  |
| **((Kite) shape6).method2();** |  |
| **((Kite) shape7).method2();** |  |
| **((Kite) shape1).method3();** |  |
| **((Kite) shape2).method3();** |  |
| **((Kite) shape3).method3();** |  |
| **((Kite) shape4).method3();** |  |
| **((Kite) shape5).method3();** |  |
| **((Kite) shape6).method3();** |  |
| **((Kite) shape7).method3();** |  |
| **((Parallelogram) shape1).method1();** |  |
| **((Parallelogram) shape2).method1();** |  |
| **((Parallelogram) shape3).method1();** |  |
| **((Parallelogram) shape4).method1();** |  |
| **((Parallelogram) shape5).method1();** |  |
| **((Parallelogram) shape6).method1();** |  |
| **((Parallelogram) shape7).method1();** |  |
| **((Parallelogram) shape1).method2();** |  |
| **((Parallelogram) shape2).method2();** |  |
| **((Parallelogram) shape3).method2();** |  |
| **((Parallelogram) shape4).method2();** |  |
| **((Parallelogram) shape5).method2();** |  |
| **((Parallelogram) shape6).method2();** |  |
| **((Parallelogram) shape7).method2();** |  |
| **((Parallelogram) shape1).method3();** |  |
| **((Parallelogram) shape2).method3();** |  |
| **((Parallelogram) shape3).method3();** |  |
| **((Parallelogram) shape4).method3();** |  |
| **((Parallelogram) shape5).method3();** |  |
| **((Parallelogram) shape6).method3();** |  |
| **((Parallelogram) shape7).method3();** |  |
| **((Trapezium) shape1).method1();** |  |
| **((Trapezium) shape2).method1();** |  |
| **((Trapezium) shape3).method1();** |  |
| **((Trapezium) shape4).method1();** |  |
| **((Trapezium) shape5).method1();** |  |
| **((Trapezium) shape6).method1();** |  |
| **((Trapezium) shape7).method1();** |  |
| **((Trapezium) shape1).method2();** |  |
| **((Trapezium) shape2).method2();** |  |
| **((Trapezium) shape3).method2();** |  |
| **((Trapezium) shape4).method2();** |  |
| **((Trapezium) shape5).method2();** |  |
| **((Trapezium) shape6).method2();** |  |
| **((Trapezium) shape7).method2();** |  |
| **((Trapezium) shape1).method3();** |  |
| **((Trapezium) shape2).method3();** |  |
| **((Trapezium) shape3).method3();** |  |
| **((Trapezium) shape4).method3();** |  |
| **((Trapezium) shape5).method3();** |  |
| **((Trapezium) shape6).method3();** |  |
| **((Trapezium) shape7).method3();** |  |
| **((Rhombus) shape1).method1();** |  |
| **((Rhombus) shape2).method1();** |  |
| **((Rhombus) shape3).method1();** |  |
| **((Rhombus) shape4).method1();** |  |
| **((Rhombus) shape5).method1();** |  |
| **((Rhombus) shape6).method1();** |  |
| **((Rhombus) shape7).method1();** |  |
| **((Rhombus) shape1).method2();** |  |
| **((Rhombus) shape2).method2();** |  |
| **((Rhombus) shape3).method2();** |  |
| **((Rhombus) shape4).method2();** |  |
| **((Rhombus) shape5).method2();** |  |
| **((Rhombus) shape6).method2();** |  |
| **((Rhombus) shape7).method2();** |  |
| **((Rhombus) shape1).method3();** |  |
| **((Rhombus) shape2).method3();** |  |
| **((Rhombus) shape3).method3();** |  |
| **((Rhombus) shape4).method3();** |  |
| **((Rhombus) shape5).method3();** |  |
| **((Rhombus) shape6).method3();** |  |
| **((Rhombus) shape7).method3();** |  |
| **((Rectangle) shape1).method1();** |  |
| **((Rectangle) shape2).method1();** |  |
| **((Rectangle) shape3).method1();** |  |
| **((Rectangle) shape4).method1();** |  |
| **((Rectangle) shape5).method1();** |  |
| **((Rectangle) shape6).method1();** |  |
| **((Rectangle) shape7).method1();** |  |
| **((Rectangle) shape1).method2();** |  |
| **((Rectangle) shape2).method2();** |  |
| **((Rectangle) shape3).method2();** |  |
| **((Rectangle) shape4).method2();** |  |
| **((Rectangle) shape5).method2();** |  |
| **((Rectangle) shape6).method2();** |  |
| **((Rectangle) shape7).method2();** |  |
| **((Rectangle) shape1).method3();** |  |
| **((Rectangle) shape2).method3();** |  |
| **((Rectangle) shape3).method3();** |  |
| **((Rectangle) shape4).method3();** |  |
| **((Rectangle) shape5).method3();** |  |
| **((Rectangle) shape6).method3();** |  |
| **((Rectangle) shape7).method3();** |  |
| **((Square) shape1).method1();** |  |
| **((Square) shape2).method1();** |  |
| **((Square) shape3).method1();** |  |
| **((Square) shape4).method1();** |  |
| **((Square) shape5).method1();** |  |
| **((Square) shape6).method1();** |  |
| **((Square) shape7).method1();** |  |
| **((Square) shape1).method2();** |  |
| **((Square) shape2).method2();** |  |
| **((Square) shape3).method2();** |  |
| **((Square) shape4).method2();** |  |
| **((Square) shape5).method2();** |  |
| **((Square) shape6).method2();** |  |
| **((Square) shape7).method2();** |  |
| **((Square) shape1).method3();** |  |
| **((Square) shape2).method3();** |  |
| **((Square) shape3).method3();** |  |
| **((Square) shape4).method3();** |  |
| **((Square) shape5).method3();** |  |
| **((Square) shape6).method3();** |  |
| **((Square) shape7).method3();** |  |