Java

Section Id :	64065339715
Section Number :	9
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	16
Number of Questions to be attempted :	16
Section Marks :	50
Display Number Panel :	Yes
Group All Questions :	No
Enable Mark as Answered Mark for Review and Clear Response :	Yes
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	64065384387
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 124 Question Id : 640653587032 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0

Question Label : Multiple Choice Question

THIS IS QUESTION PAPER FOR THE SUBJECT "DIPLOMA LEVEL : PROGRAMMING CONCEPTS USING JAVA (COMPUTER BASED EXAM)"

ARE YOU SURE YOU HAVE TO WRITE EXAM FOR THIS SUBJECT?
CROSS CHECK YOUR HALL TICKET TO CONFIRM THE SUBJECTS TO BE WRITTEN.

(IF IT IS NOT THE CORRECT SUBJECT, PLS CHECK THE SECTION AT THE TOP FOR THE SUBJECTS REGISTERED BY YOU)

Options :

6406531958770. ✓ YES

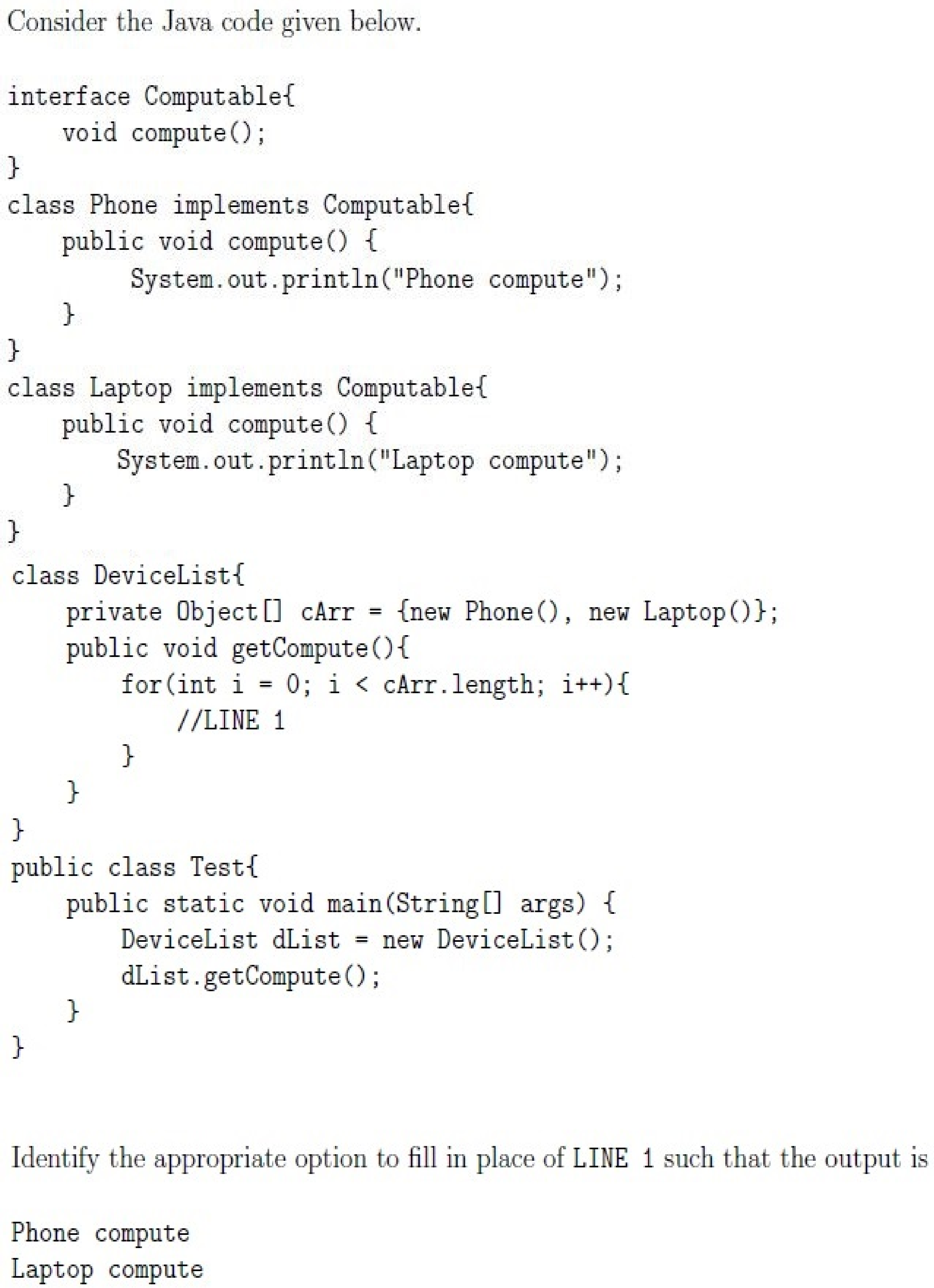
6406531958771. ✖ NO

Sub-Section Number :	2
Sub-Section Id :	64065384388
Question Shuffling Allowed :	Yes
Is Section Default? :	null

Question Number : 125 Question Id : 640653587033 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



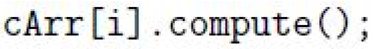
Consider the Java code given below.

```
interface Computable{
    void compute();
}
class Phone implements Computable{
    public void compute() {
         System.out.println("Phone compute");
    }
}
class Laptop implements Computable{
    public void compute() {
        System.out.println("Laptop compute");
    }
}
class DeviceList{
    private Object[] cArr = {new Phone(), new Laptop()};
    public void getCompute(){
        for(int i = 0; i < cArr.length; i++){
            //LINE 1
        }
    }
}
public class Test{
    public static void main(String[] args) {
        DeviceList dList = new DeviceList();
        dList.getCompute();
    }
}
```

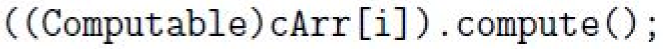
Identify the appropriate option to fill in place of LINE 1 such that the output is

```
Phone compute
Laptop compute
```

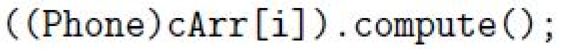
Options :

6406531958772. ✖ 

`cArr[i].compute();`

6406531958773. ✓ 

`((Computable)cArr[i]).compute();`

6406531958774. ✖ 

`((Phone)cArr[i]).compute();`

6406531958775. ✖ `((Laptop)cArr[i]).compute();`

Question Number : 126 Question Id : 640653587034 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

Consider the Java code given below.

```
public class Example{
   public <T extends Comparable> void sortArray(T[] obj){
     // Sorts obj
   }
   public <T> void elementDisplay(T[] arr){
      // Displays the elements of arr
   }
   public <T extends Number> T sum(List<T> lst){
      // Returns the sum of elements of lst
   }
}
```

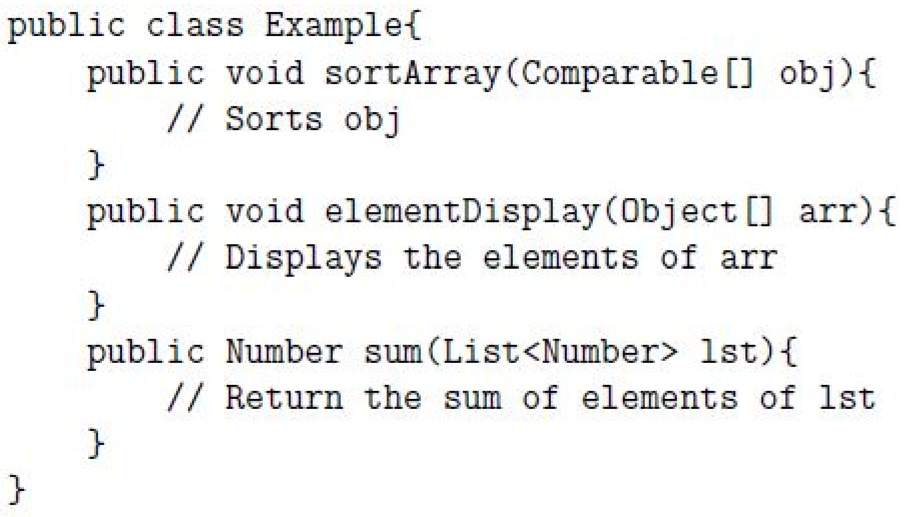
How does class Example look after type erasure?

Options :

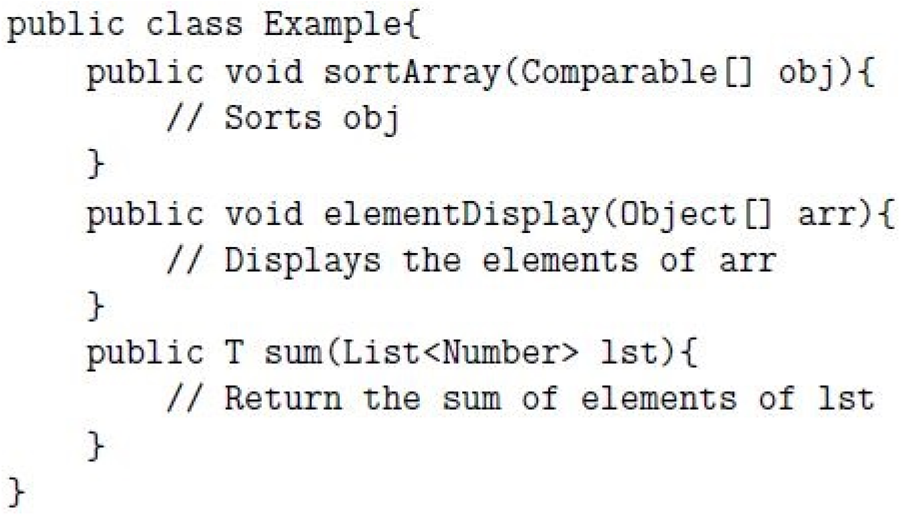
6406531958776. ✖

```
public class Example{
    public void sortArray(Object[] obj){
        // Sorts obj
    }
    public void elementDisplay(Object[] arr){
        // Displays the elements of arr
    }
    public Object sum(List<Object> lst){
        // Returns the sum of elements of lst
    }
}
```

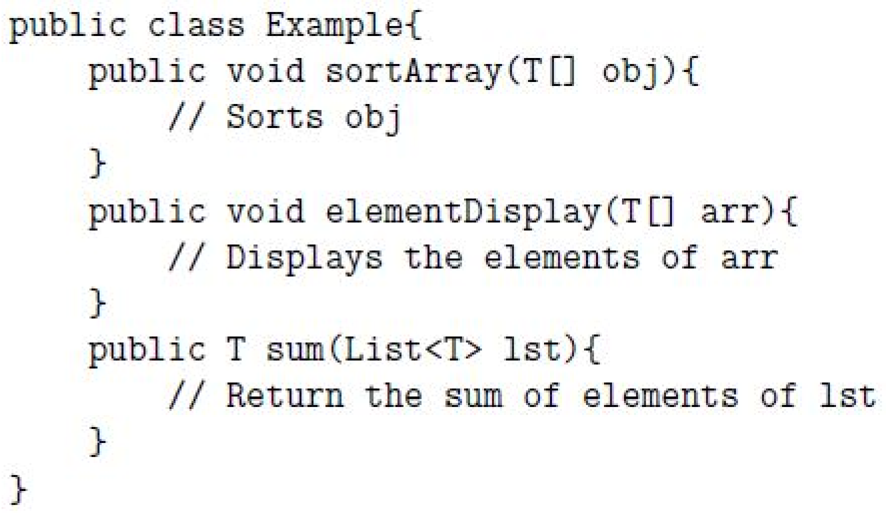
6406531958777. ✓



```
public class Example{
    public void sortArray(Comparable[] obj){
        // Sorts obj
    }
    public void elementDisplay(Object[] arr){
        // Displays the elements of arr
    }
    public Number sum(List<Number> lst){
        // Return the sum of elements of lst
    }
}
```

6406531958778. ✖ 

```
public class Example{
    public void sortArray(Comparable[] obj){
        // Sorts obj
    }
    public void elementDisplay(Object[] arr){
        // Displays the elements of arr
    }
    public T sum(List<Number> lst){
        // Return the sum of elements of lst
    }
}
```

6406531958779. ✖ 

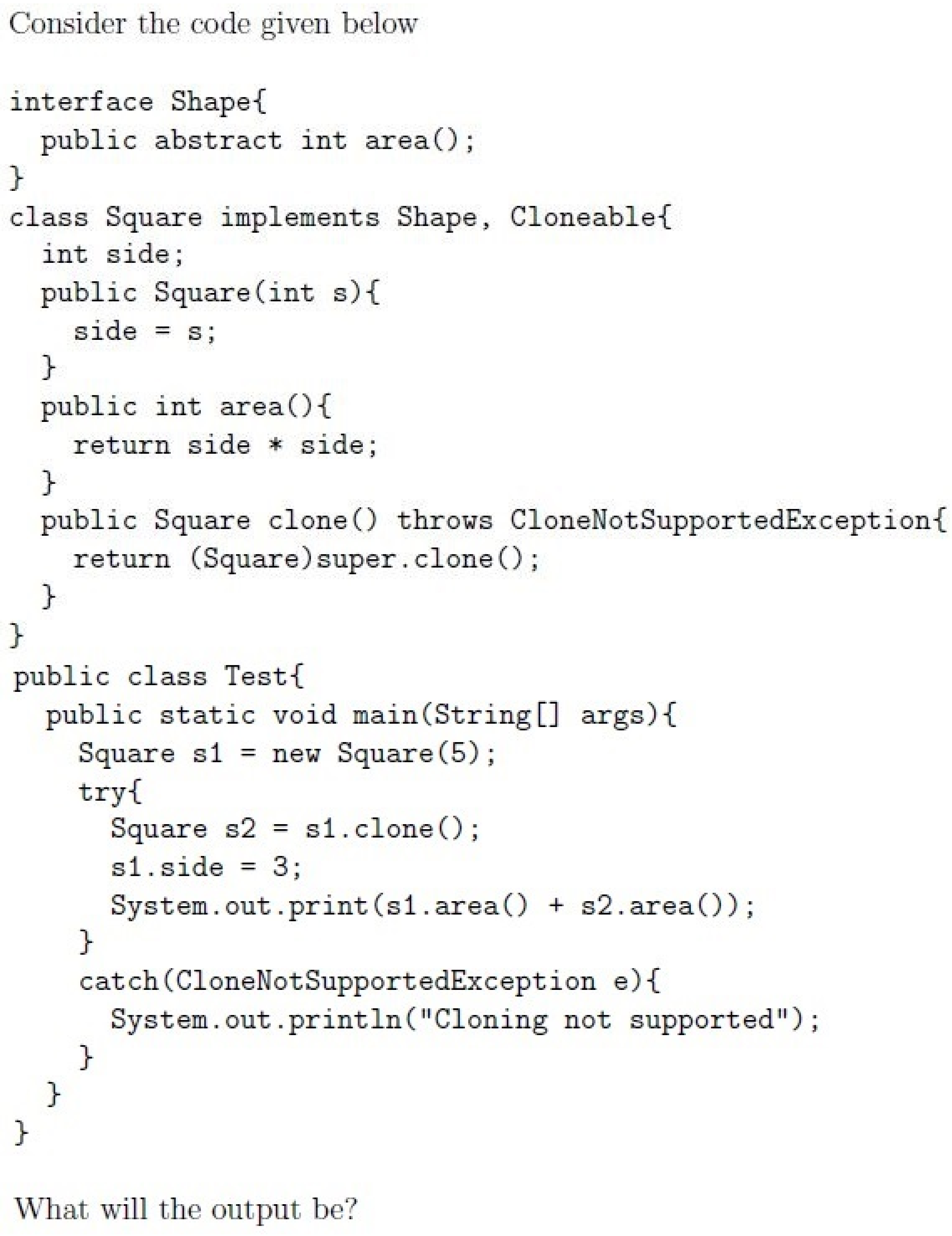
```
public class Example{
    public void sortArray(T[] obj){
        // Sorts obj
    }
    public void elementDisplay(T[] arr){
        // Displays the elements of arr
    }
    public T sum(List<T> lst){
        // Return the sum of elements of lst
    }
}
```

Question Number : 127 Question Id : 640653587036 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction

Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



Consider the code given below

```
interface Shape{
  public abstract int area();
}
class Square implements Shape, Cloneable{
  int side;
  public Square(int s){
    side = s;
  }
  public int area(){
    return side * side;
  }
  public Square clone() throws CloneNotSupportedException{
    return (Square)super.clone();
  }
}
public class Test{
  public static void main(String[] args){
    Square s1 = new Square(5);
    try{
      Square s2 = s1.clone();
      s1.side = 3;
      System.out.print(s1.area() + s2.area());
    }
    catch(CloneNotSupportedException e){
      System.out.println("Cloning not supported");
    }
  }
}
```

What will the output be?

Options :

6406531958784. ✖ 18

6406531958785. ✖ 50

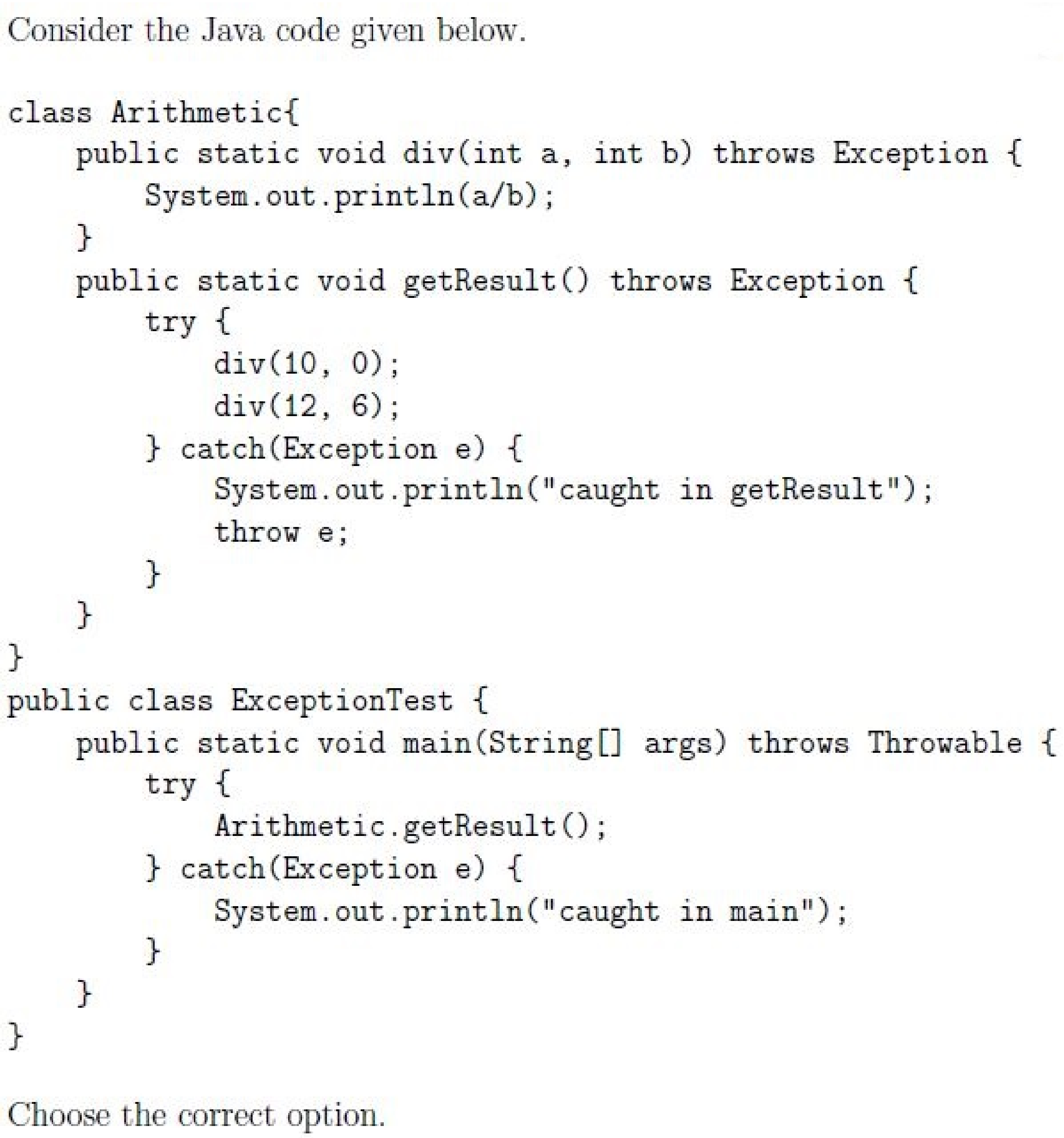
6406531958786. ✓ 34

6406531958787. ✖ Cloning not supported

Question Number : 128 Question Id : 640653587040 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



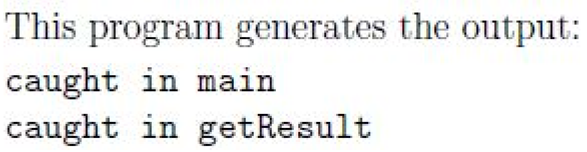
Consider the Java code given below.

```
class Arithmetic{
    public static void div(int a, int b) throws Exception {
        System.out.println(a/b);
    }
    public static void getResult() throws Exception {
        try {
            div(10, 0);
            div(12, 6);
        } catch(Exception e) {
            System.out.println("caught in getResult");
            throw e;
        }
    }
}
public class ExceptionTest {
    public static void main(String[] args) throws Throwable {
        try {
            Arithmetic.getResult();
        } catch(Exception e) {
            System.out.println("caught in main");
        }
    }
}
```

Choose the correct option.

Options :

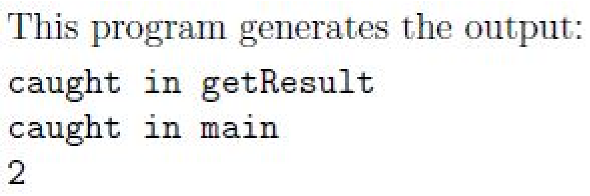
6406531958800. ✖



This program generates the output:

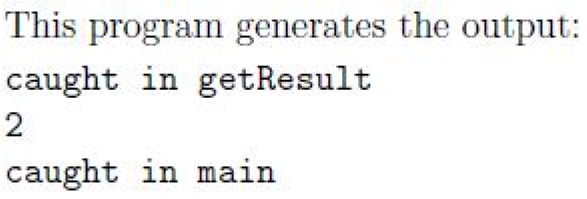
```
caught in main
caught in getResult
```

6406531958801. ✖



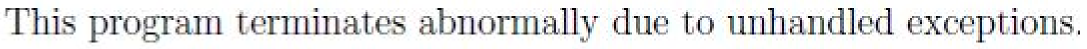
This program generates the output:

```
caught in getResult
caught in main
2
```

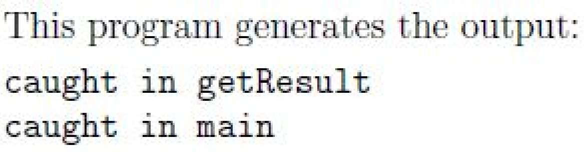
6406531958802. ✖ 

This program generates the output:

```
caught in getResult
2
caught in main
```

6406531958803. ✖ 

This program terminates abnormally due to unhandled exceptions.

6406531958804. ✓ 

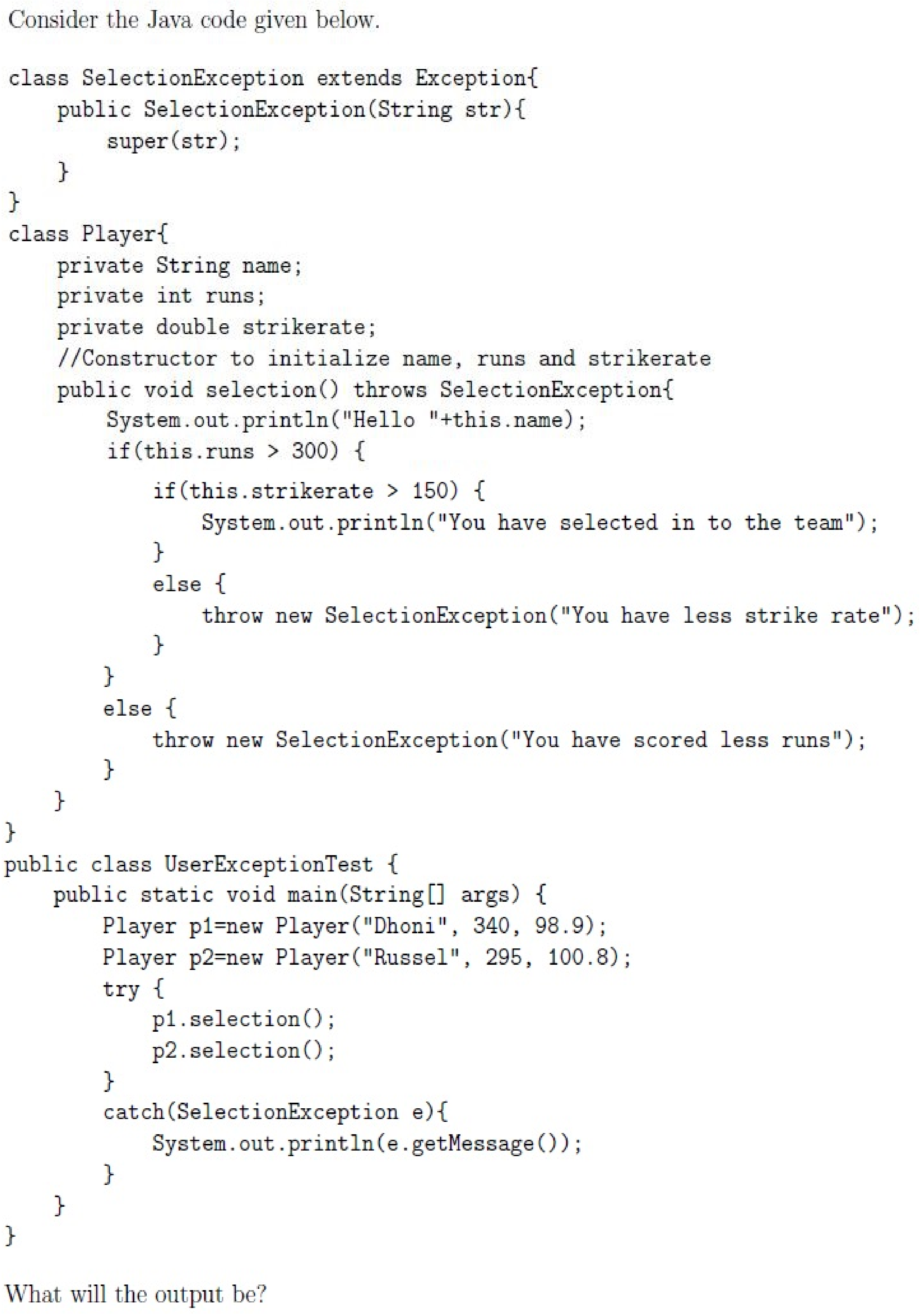
This program generates the output:

```
caught in getResult
caught in main
```

Question Number : 129 Question Id : 640653587041 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

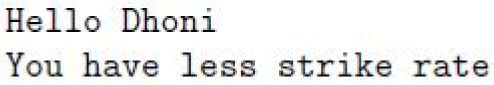


Consider the Java code given below.

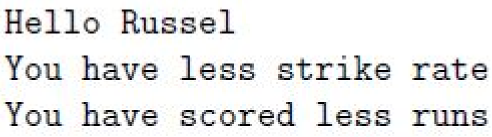
```
class SelectionException extends Exception{
    public SelectionException(String str){
        super(str);
    }
}
class Player{
    private String name;
    private int runs;
    private double strikerate;
    //Constructor to initialize name, runs and strikerate
    public void selection() throws SelectionException{
        System.out.println("Hello "+this.name);
        if(this.runs > 300) {
            if(this.strikerate > 150) {
                System.out.println("You have selected in to the team");
            }
            else {
                throw new SelectionException("You have less strike rate");
            }
        }
        else {
            throw new SelectionException("You have scored less runs");
        }
    }
}
public class UserExceptionTest {
    public static void main(String[] args) {
        Player p1=new Player("Dhoni", 340, 98.9);
        Player p2=new Player("Russel", 295, 100.8);
        try {
            p1.selection();
            p2.selection();
        }
        catch(SelectionException e){
            System.out.println(e.getMessage());
        }
    }
}
```

What will the output be?

Options :

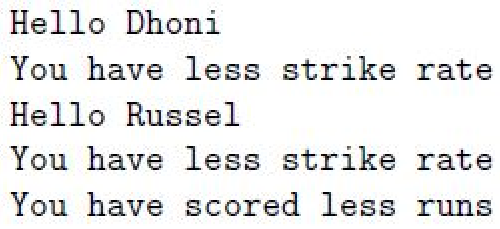
6406531958805. ✓ 

```
Hello Dhoni
You have less strike rate
```

6406531958806. ✗ 

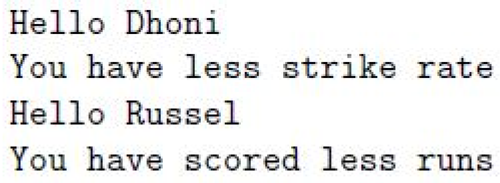
```
Hello Russel
You have less strike rate
You have scored less runs
```

6406531958807. ✖



```
Hello Dhoni
You have less strike rate
Hello Russel
You have less strike rate
You have scored less runs
```

6406531958808. ✖

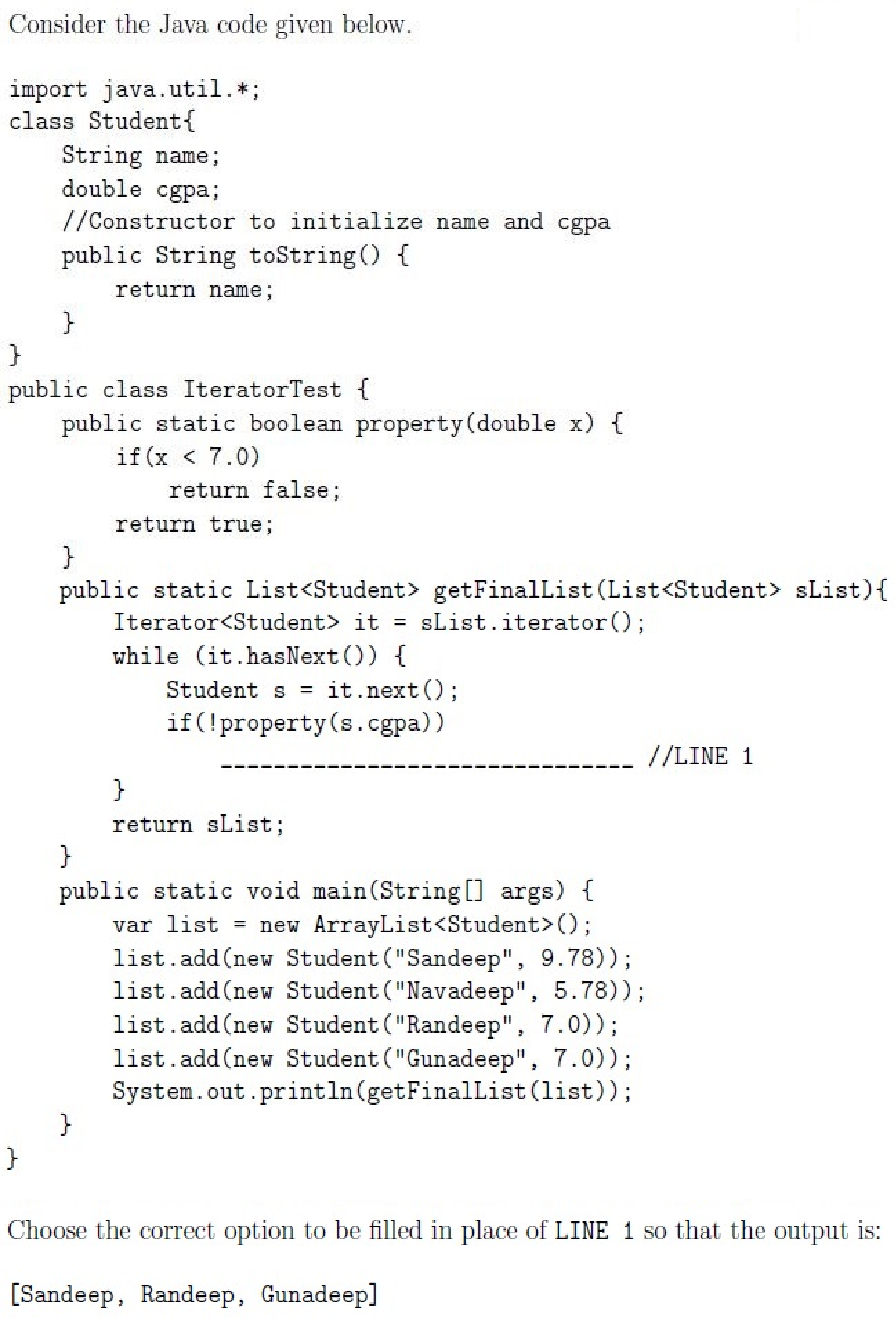


```
Hello Dhoni
You have less strike rate
Hello Russel
You have scored less runs
```

Question Number : 130 Question Id : 640653587044 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



Consider the Java code given below.

```
import java.util.*;
class Student{
    String name;
    double cgpa;
    //Constructor to initialize name and cgpa
    public String toString() {
        return name;
    }
}
public class IteratorTest {
    public static boolean property(double x) {
        if(x < 7.0)
            return false;
        return true;
    }
    public static List<Student> getFinalList(List<Student> sList){
        Iterator<Student> it = sList.iterator();
        while (it.hasNext()) {
            Student s = it.next();
            if(!property(s.cgpa))
                _______________________________ //LINE 1
        }
        return sList;
    }
    public static void main(String[] args) {
        var list = new ArrayList<Student>();
        list.add(new Student("Sandeep", 9.78));
        list.add(new Student("Navadeep", 5.78));
        list.add(new Student("Randeep", 7.0));
        list.add(new Student("Gunadeep", 7.0));
        System.out.println(getFinalList(list));
    }
}
```

Choose the correct option to be filled in place of `LINE 1` so that the output is:

```
[Sandeep, Randeep, Gunadeep]
```

Options :

6406531958818. ✗ `it.remove(s)`

6406531958819. ✓ `it.remove()`

6406531958820. ✖

`sList.remove()`

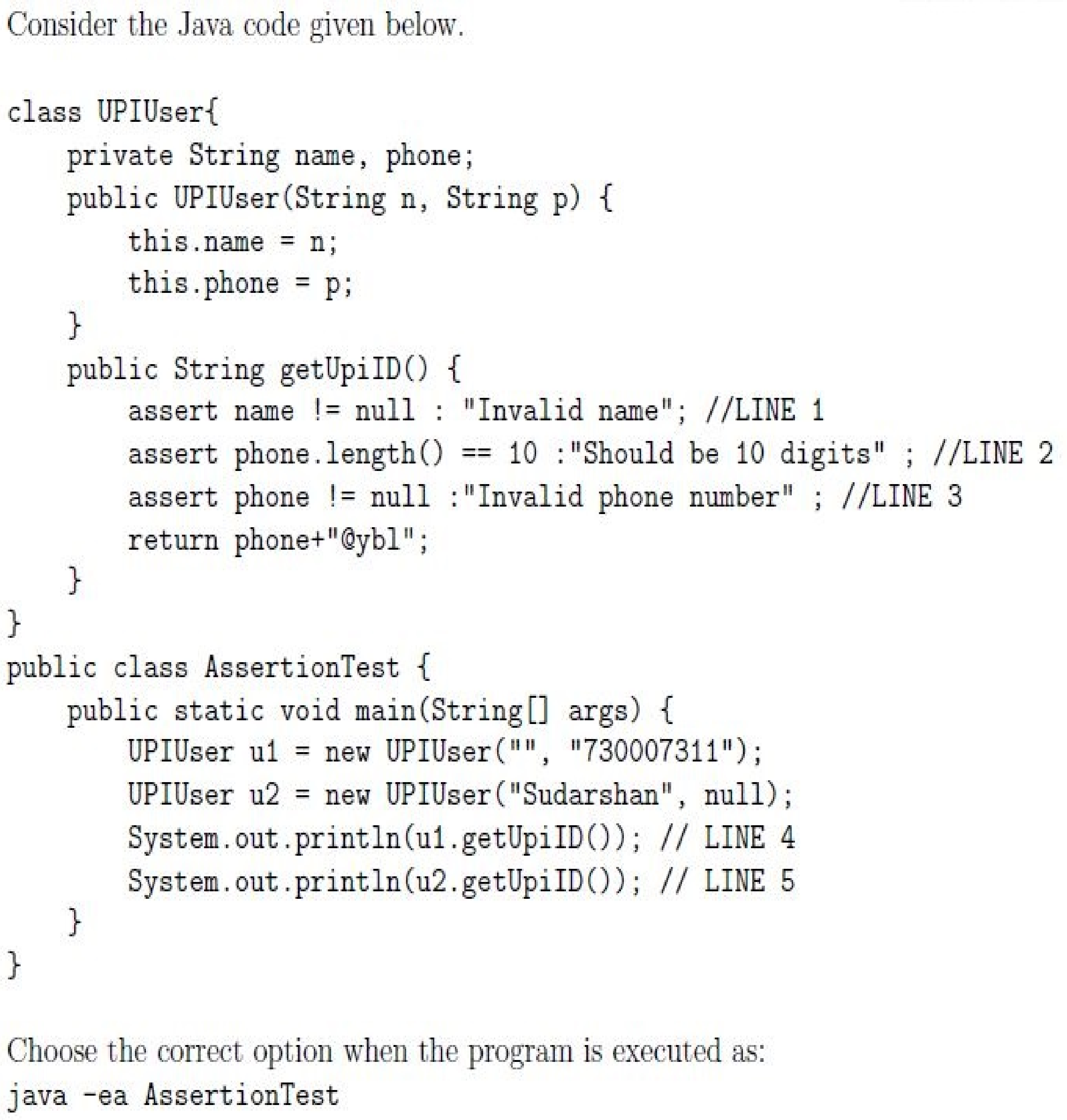
6406531958821. ✖

`sList.remove(s)`

Question Number : 131 Question Id : 640653587045 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



Consider the Java code given below.

```
class UPIUser{
    private String name, phone;
    public UPIUser(String n, String p) {
        this.name = n;
        this.phone = p;
    }
    public String getUpiID() {
        assert name != null : "Invalid name"; //LINE 1
        assert phone.length() == 10 :"Should be 10 digits" ; //LINE 2
        assert phone != null :"Invalid phone number" ; //LINE 3
        return phone+"@ybl";
    }
}
public class AssertionTest {
    public static void main(String[] args) {
        UPIUser u1 = new UPIUser("", "730007311");
        UPIUser u2 = new UPIUser("Sudarshan", null);
        System.out.println(u1.getUpiID()); // LINE 4
        System.out.println(u2.getUpiID()); // LINE 5
    }
}
```

Choose the correct option when the program is executed as:

```
java -ea AssertionTest
```

Options :

6406531958822. ✖

LINE 1 throws AssertionError when LINE 4 is executed.

6406531958823. ✓ LINE 2 throws AssertionError when LINE 4 is executed.

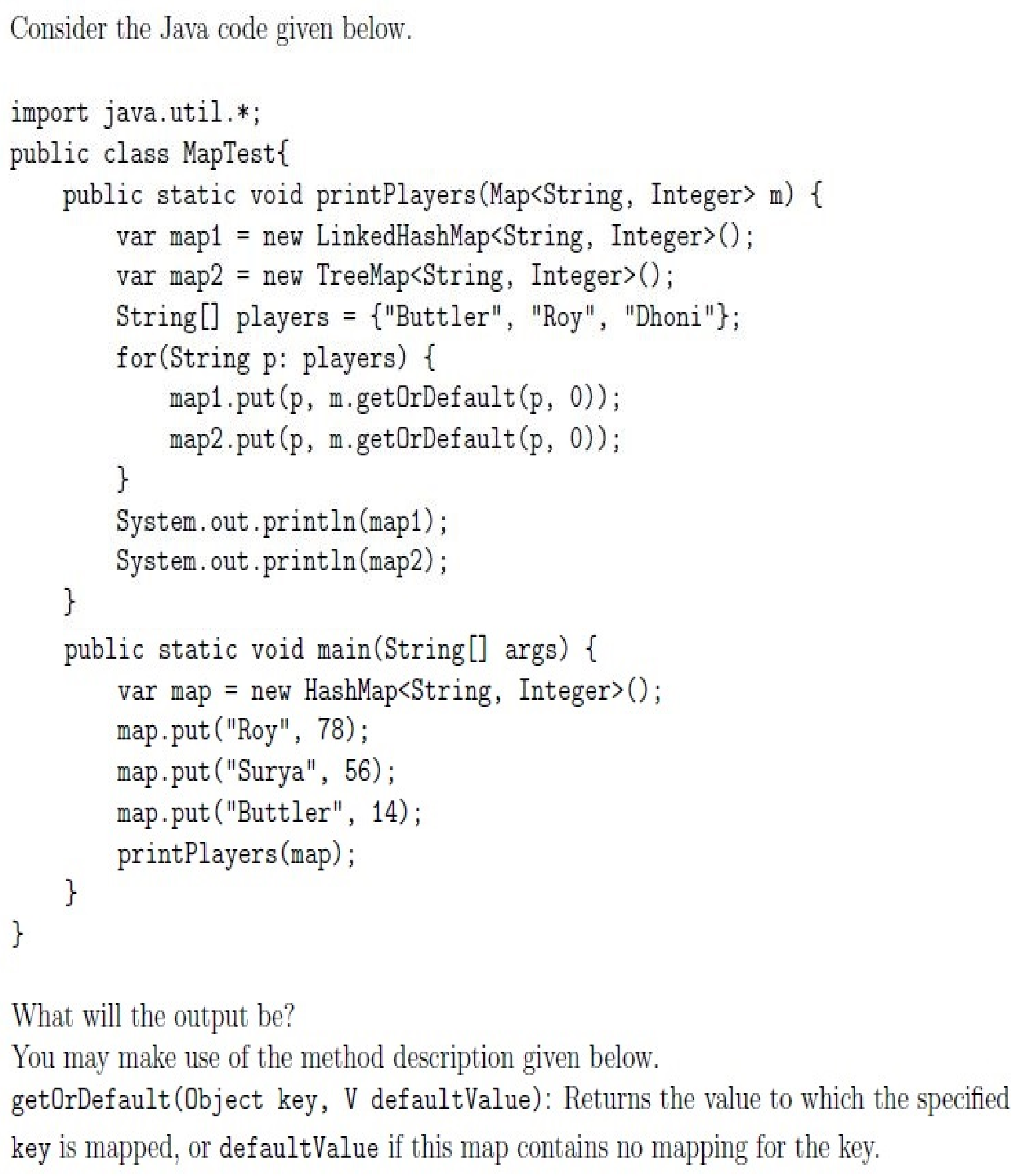
6406531958824. ✖ LINE 2 throws AssertionError when LINE 5 is executed.

6406531958825. ✖ LINE 3 throws AssertionError when LINE 5 is executed.

Question Number : 132 Question Id : 640653587046 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question

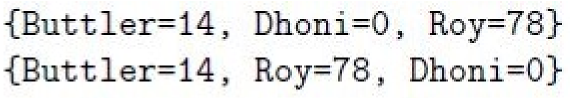


Consider the Java code given below.

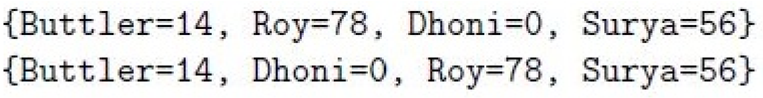
```
import java.util.*;
public class MapTest{
    public static void printPlayers(Map<String, Integer> m) {
        var map1 = new LinkedHashMap<String, Integer>();
        var map2 = new TreeMap<String, Integer>();
        String[] players = {"Buttler", "Roy", "Dhoni"};
        for(String p: players) {
            map1.put(p, m.getOrDefault(p, 0));
            map2.put(p, m.getOrDefault(p, 0));
        }
        System.out.println(map1);
        System.out.println(map2);
    }
    public static void main(String[] args) {
        var map = new HashMap<String, Integer>();
        map.put("Roy", 78);
        map.put("Surya", 56);
        map.put("Buttler", 14);
        printPlayers(map);
    }
}
```

What will the output be?
You may make use of the method description given below.
`getOrDefault(Object key, V defaultValue)`: Returns the value to which the specified `key` is mapped, or `defaultValue` if this map contains no mapping for the key.

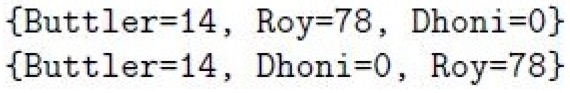
Options :

6406531958826. ✖ 

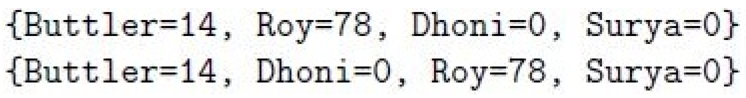
```
{Buttler=14, Dhoni=0, Roy=78}
{Buttler=14, Roy=78, Dhoni=0}
```

6406531958827. ✖ 

```
{Buttler=14, Roy=78, Dhoni=0, Surya=56}
{Buttler=14, Dhoni=0, Roy=78, Surya=56}
```

6406531958828. ✓ 

```
{Buttler=14, Roy=78, Dhoni=0}
{Buttler=14, Dhoni=0, Roy=78}
```

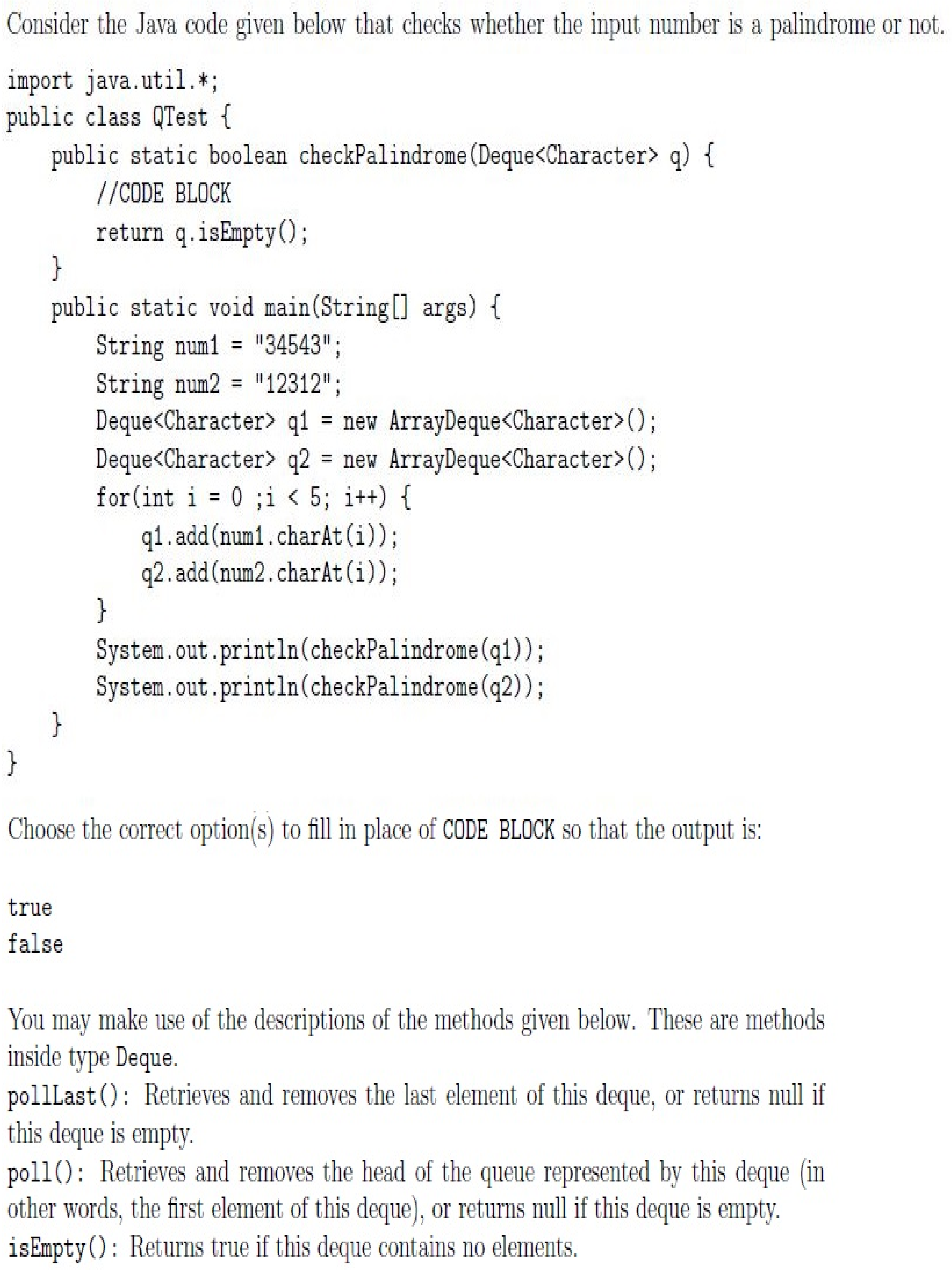
6406531958829. ✖ 

```
{Buttler=14, Roy=78, Dhoni=0, Surya=0}
{Buttler=14, Dhoni=0, Roy=78, Surya=0}
```

Question Number : 133 Question Id : 640653587047 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3

Question Label : Multiple Choice Question



Consider the Java code given below that checks whether the input number is a palindrome or not.

```
import java.util.*;
public class QTest {
    public static boolean checkPalindrome(Deque<Character> q) {
        //CODE BLOCK
        return q.isEmpty();
    }
    public static void main(String[] args) {
        String num1 = "34543";
        String num2 = "12312";
        Deque<Character> q1 = new ArrayDeque<Character>();
        Deque<Character> q2 = new ArrayDeque<Character>();
        for(int i = 0 ;i < 5; i++) {
            q1.add(num1.charAt(i));
            q2.add(num2.charAt(i));
        }
        System.out.println(checkPalindrome(q1));
        System.out.println(checkPalindrome(q2));
    }
}
```

Choose the correct option(s) to fill in place of `CODE BLOCK` so that the output is:

```
true
false
```

You may make use of the descriptions of the methods given below. These are methods inside type `Deque`.

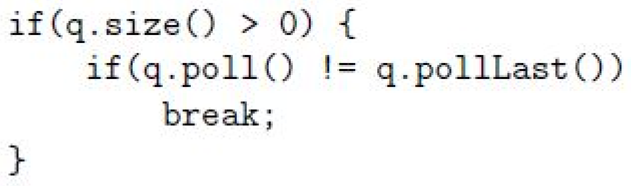
`pollLast()`: Retrieves and removes the last element of this deque, or returns null if this deque is empty.

`poll()`: Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty.

`isEmpty()`: Returns true if this deque contains no elements.

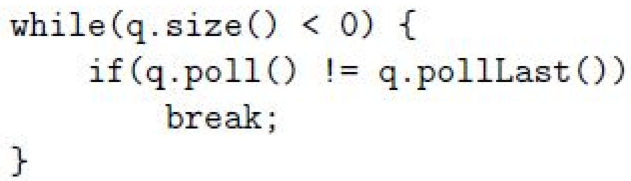
Options :

6406531958830. ✖



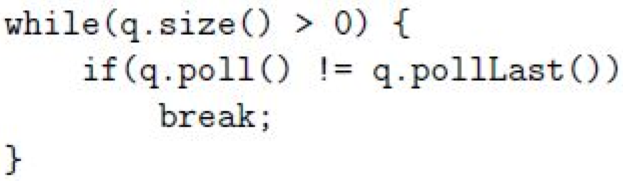
```
if(q.size() > 0) {
    if(q.poll() != q.pollLast())
        break;
}
```

6406531958831. ✖



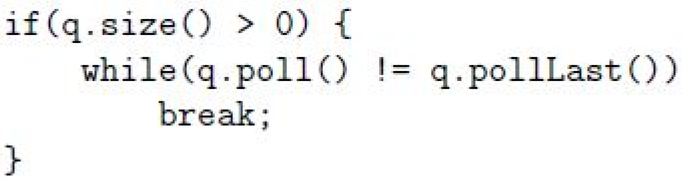
```
while(q.size() < 0) {
    if(q.poll() != q.pollLast())
        break;
}
```

6406531958832. ✓



```
while(q.size() > 0) {
    if(q.poll() != q.pollLast())
        break;
}
```

6406531958833. ✗



```
if(q.size() > 0) {
    while(q.poll() != q.pollLast())
        break;
}
```

Sub-Section Number : 3

Sub-Section Id : 64065384389

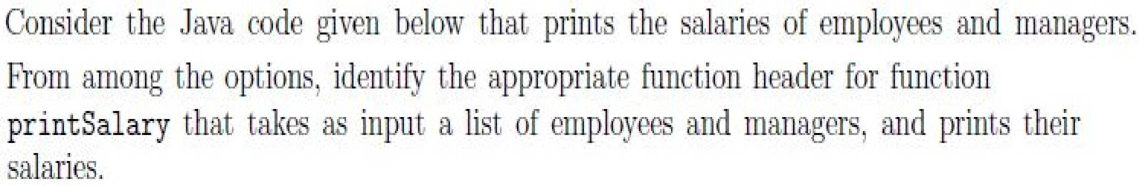
Question Shuffling Allowed : Yes

Is Section Default? : null

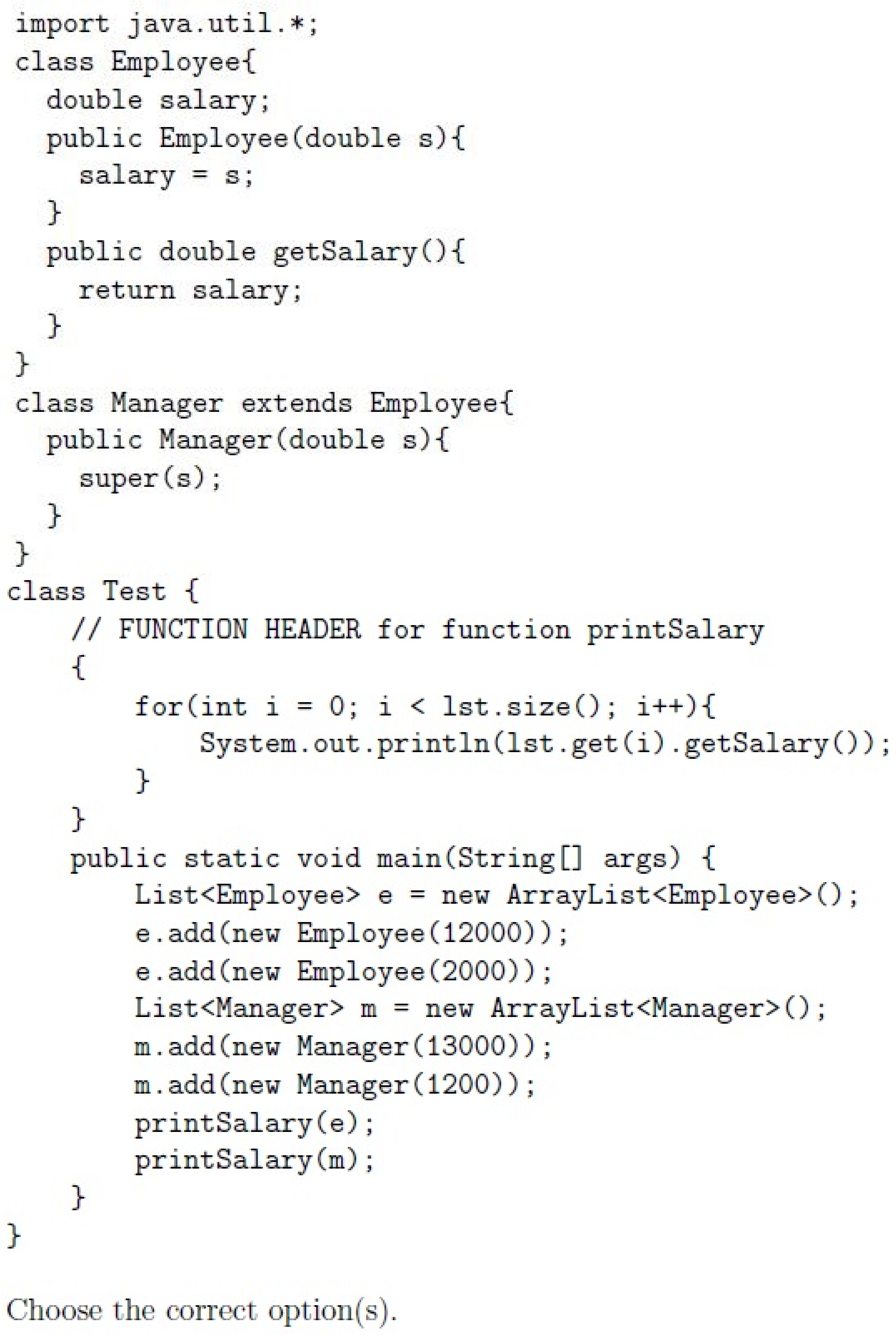
Question Number : 134 Question Id : 640653587035 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4 Max. Selectable Options : 0

Question Label : Multiple Select Question



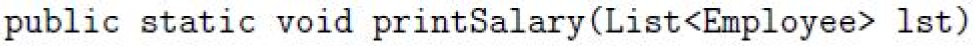
Consider the Java code given below that prints the salaries of employees and managers. From among the options, identify the appropriate function header for function `printSalary` that takes as input a list of employees and managers, and prints their salaries.



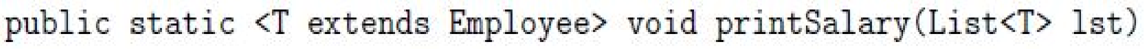
```
import java.util.*;
class Employee{
  double salary;
  public Employee(double s){
    salary = s;
  }
  public double getSalary(){
    return salary;
  }
}
class Manager extends Employee{
  public Manager(double s){
    super(s);
  }
}
class Test {
    // FUNCTION HEADER for function printSalary
    {
        for(int i = 0; i < lst.size(); i++){
            System.out.println(lst.get(i).getSalary());
        }
    }
    public static void main(String[] args) {
        List<Employee> e = new ArrayList<Employee>();
        e.add(new Employee(12000));
        e.add(new Employee(2000));
        List<Manager> m = new ArrayList<Manager>();
        m.add(new Manager(13000));
        m.add(new Manager(1200));
        printSalary(e);
        printSalary(m);
    }
}
```

Choose the correct option(s).

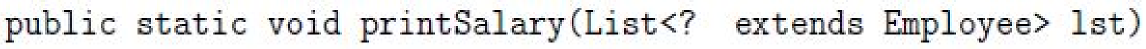
Options :

6406531958780. ✗ 

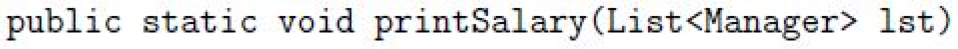
`public static void printSalary(List<Employee> lst)`

6406531958781. ✓ 

`public static <T extends Employee> void printSalary(List<T> lst)`

6406531958782. ✓ 

`public static void printSalary(List<?  extends Employee> lst)`

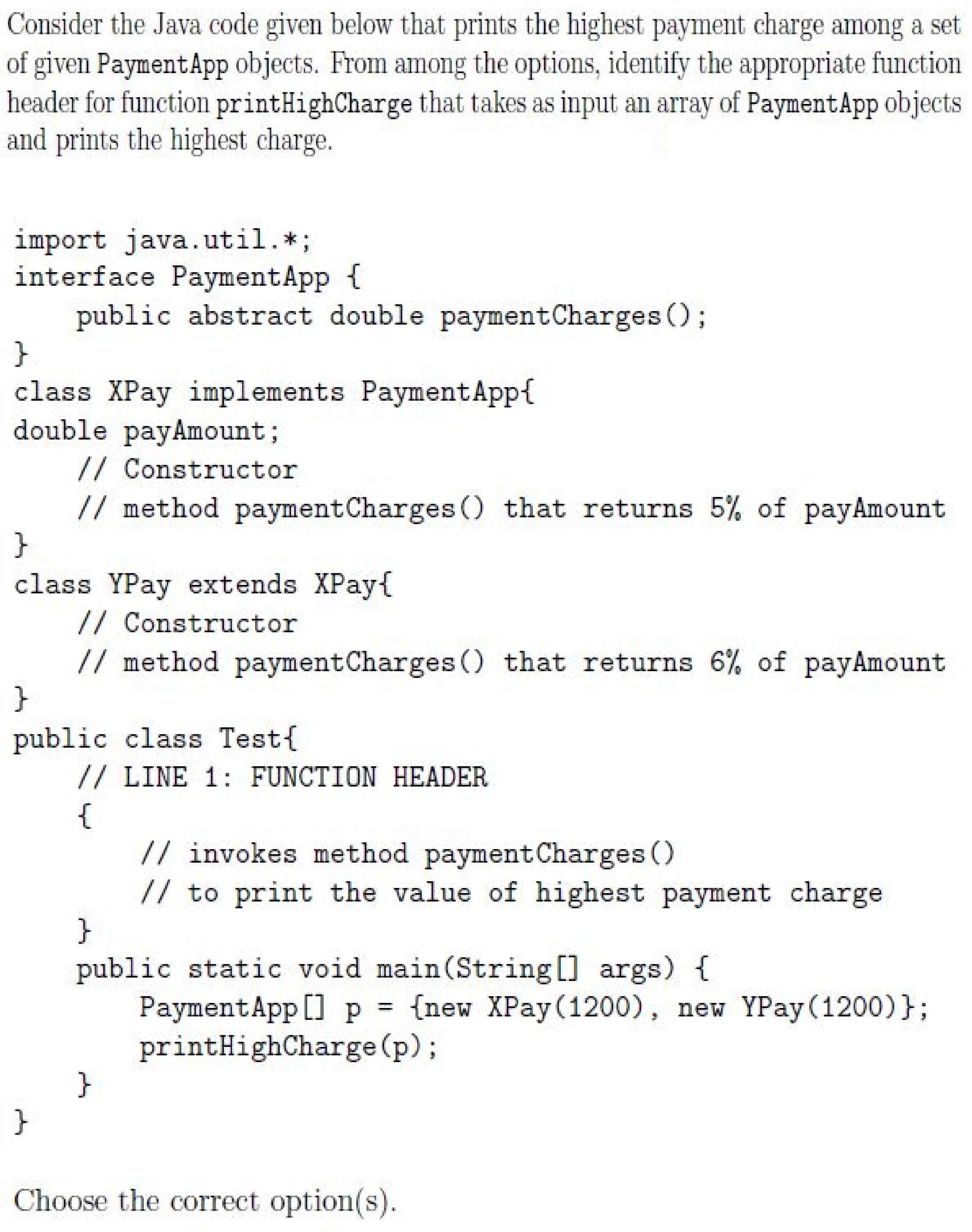
6406531958783. ✖ 

```
public static void printSalary(List<Manager> lst)
```

Question Number : 135 Question Id : 640653587037 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4 Max. Selectable Options : 0

Question Label : Multiple Select Question

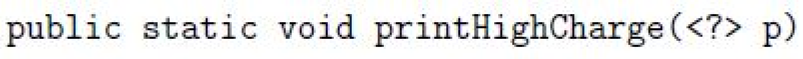


Consider the Java code given below that prints the highest payment charge among a set of given PaymentApp objects. From among the options, identify the appropriate function header for function printHighCharge that takes as input an array of PaymentApp objects and prints the highest charge.

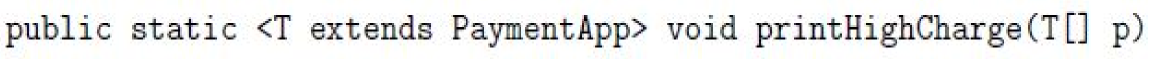
```
import java.util.*;
interface PaymentApp {
    public abstract double paymentCharges();
}
class XPay implements PaymentApp{
double payAmount;
    // Constructor
    // method paymentCharges() that returns 5% of payAmount
}
class YPay extends XPay{
    // Constructor
    // method paymentCharges() that returns 6% of payAmount
}
public class Test{
    // LINE 1: FUNCTION HEADER
    {
        // invokes method paymentCharges()
        // to print the value of highest payment charge
    }
    public static void main(String[] args) {
        PaymentApp[] p = {new XPay(1200), new YPay(1200)};
        printHighCharge(p);
    }
}
```

Choose the correct option(s).

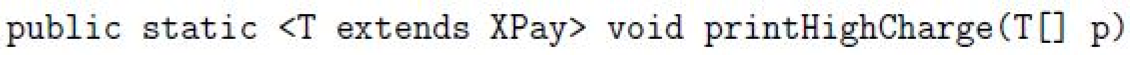
Options :

6406531958788. ✖ 

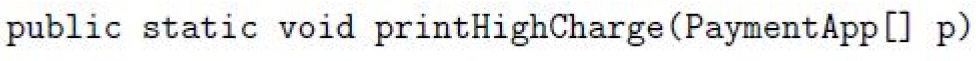
```
public static void printHighCharge(<?> p)
```

6406531958789. ✓ 

```
public static <T extends PaymentApp> void printHighCharge(T[] p)
```

6406531958790. ✖ 

```
public static <T extends XPay> void printHighCharge(T[] p)
```

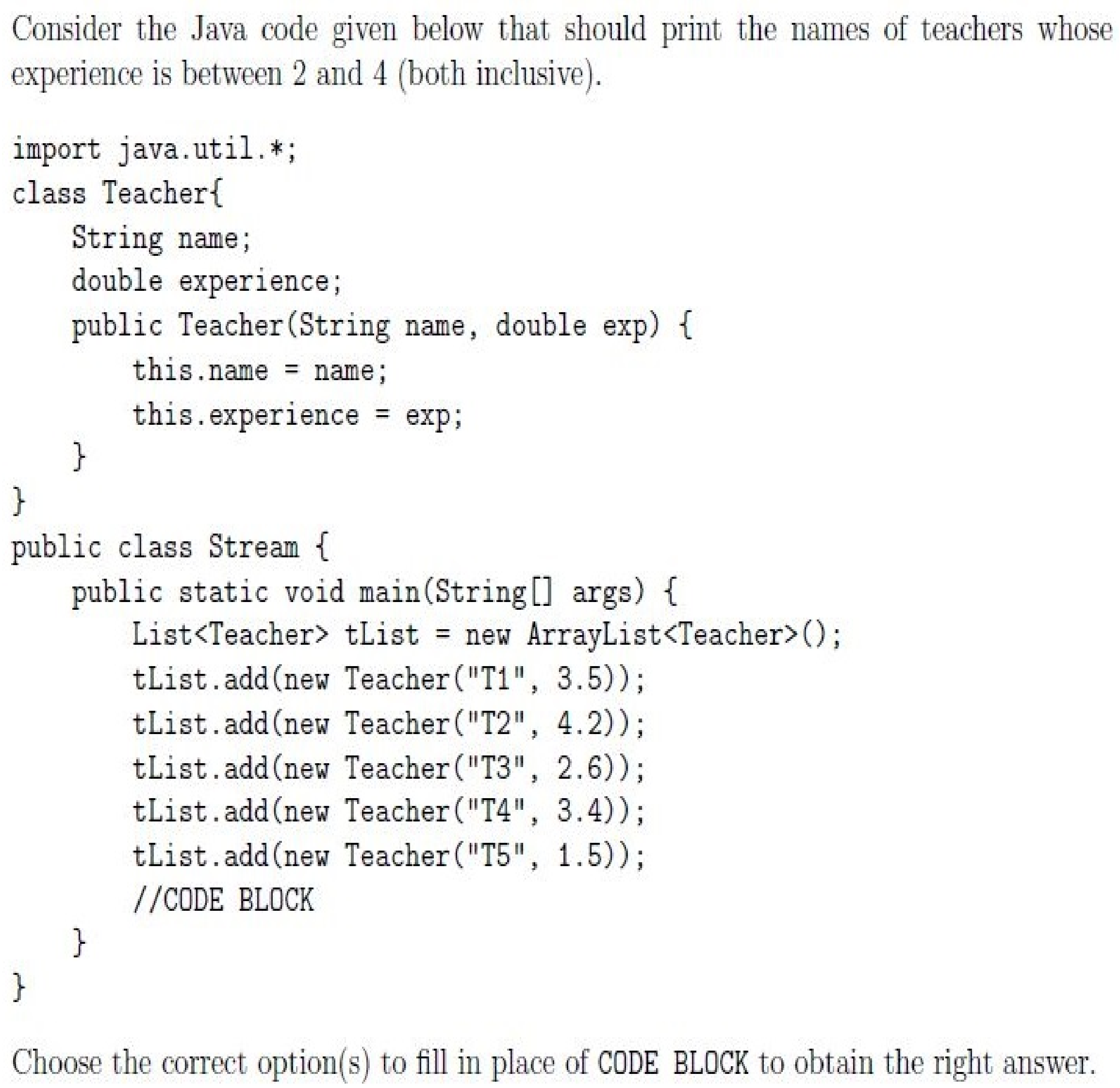
6406531958791. ✓ 

```
public static void printHighCharge(PaymentApp[] p)
```

Question Number : 136 Question Id : 640653587038 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4 Max. Selectable Options : 0

Question Label : Multiple Select Question



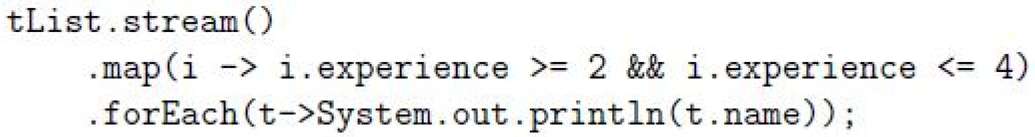
Consider the Java code given below that should print the names of teachers whose experience is between 2 and 4 (both inclusive).

```
import java.util.*;
class Teacher{
    String name;
    double experience;
    public Teacher(String name, double exp) {
        this.name = name;
        this.experience = exp;
    }
}
public class Stream {
    public static void main(String[] args) {
        List<Teacher> tList = new ArrayList<Teacher>();
        tList.add(new Teacher("T1", 3.5));
        tList.add(new Teacher("T2", 4.2));
        tList.add(new Teacher("T3", 2.6));
        tList.add(new Teacher("T4", 3.4));
        tList.add(new Teacher("T5", 1.5));
        //CODE BLOCK
    }
}
```

Choose the correct option(s) to fill in place of CODE BLOCK to obtain the right answer.

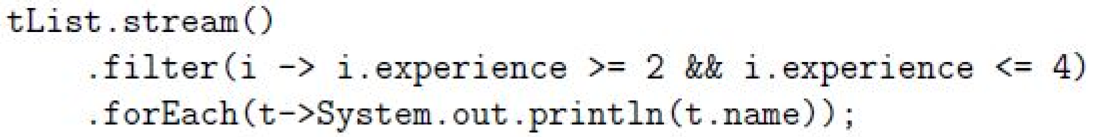
Options :

6406531958792. ✗



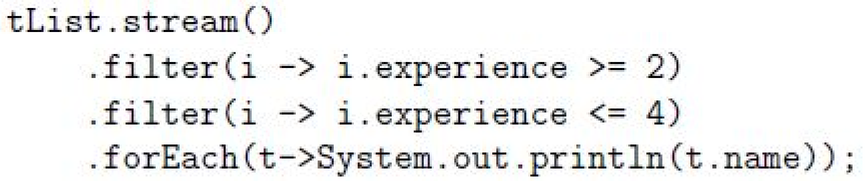
```
tList.stream()
    .map(i -> i.experience >= 2 && i.experience <= 4)
    .forEach(t->System.out.println(t.name));
```

6406531958793. ✓



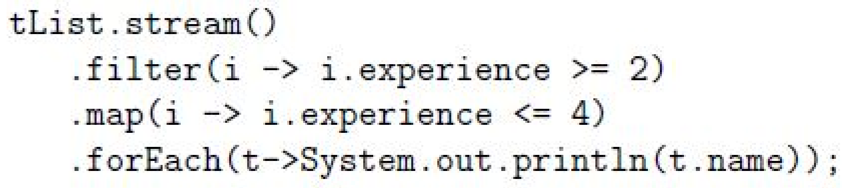
```
tList.stream()
    .filter(i -> i.experience >= 2 && i.experience <= 4)
    .forEach(t->System.out.println(t.name));
```

6406531958794. ✓



```
tList.stream()
    .filter(i -> i.experience >= 2)
    .filter(i -> i.experience <= 4)
    .forEach(t->System.out.println(t.name));
```

6406531958795. ✗



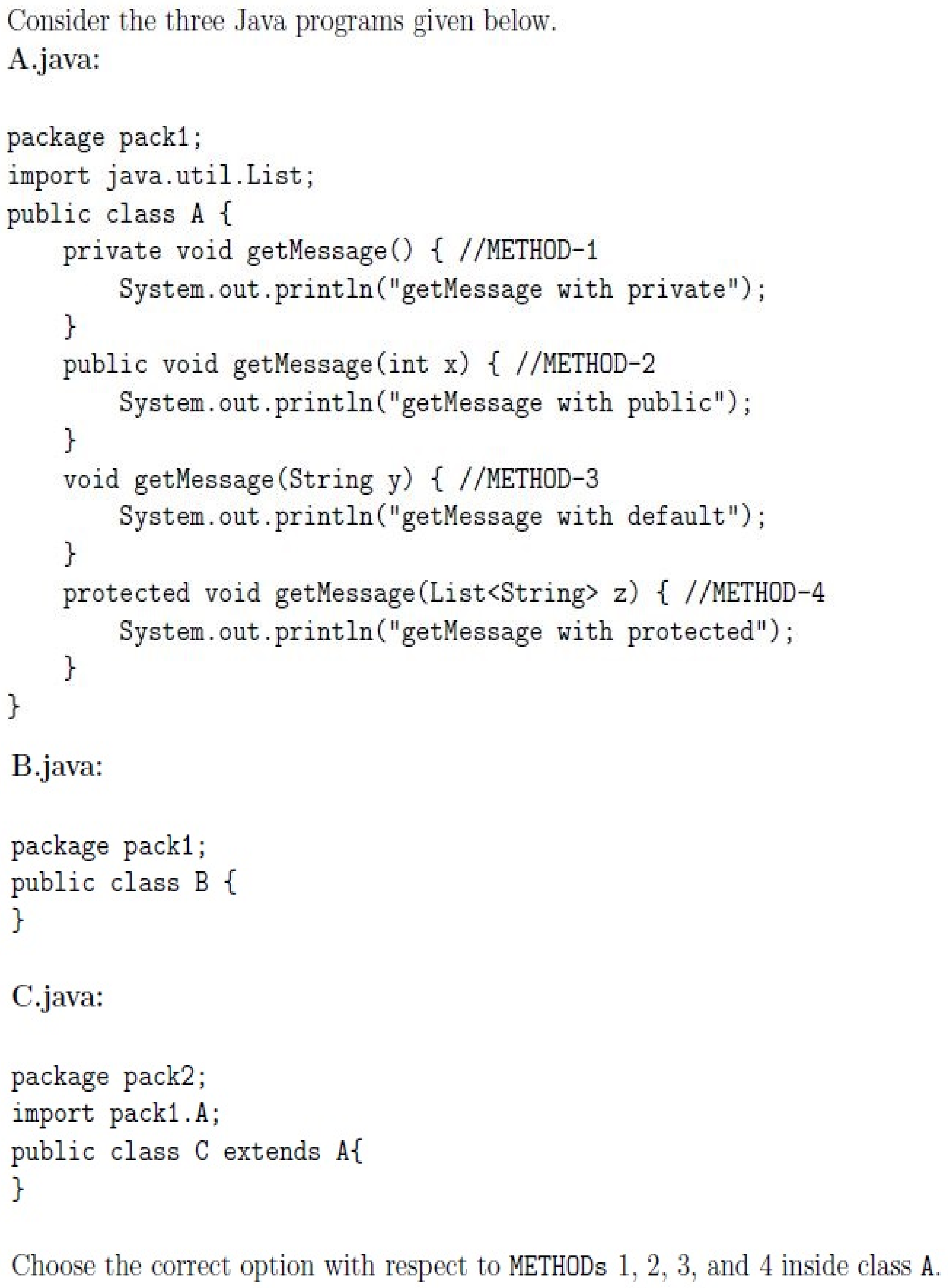
```
tList.stream()
    .filter(i -> i.experience >= 2)
    .map(i -> i.experience <= 4)
    .forEach(t->System.out.println(t.name));
```

Sub-Section Number :	4
Sub-Section Id :	64065384390
Question Shuffling Allowed :	Yes
Is Section Default? :	null

Question Number : 137 Question Id : 640653587042 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 3 Max. Selectable Options : 0

Question Label : Multiple Select Question



Consider the three Java programs given below.

A.java:

```
package pack1;
import java.util.List;
public class A {
    private void getMessage() { //METHOD-1
        System.out.println("getMessage with private");
    }
    public void getMessage(int x) { //METHOD-2
        System.out.println("getMessage with public");
    }
    void getMessage(String y) { //METHOD-3
        System.out.println("getMessage with default");
    }
    protected void getMessage(List<String> z) { //METHOD-4
        System.out.println("getMessage with protected");
    }
}
```

B.java:

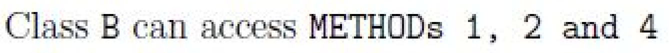
```
package pack1;
public class B {
}
```

C.java:

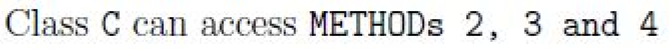
```
package pack2;
import pack1.A;
public class C extends A{
}
```

Choose the correct option with respect to METHODs 1, 2, 3, and 4 inside class A.

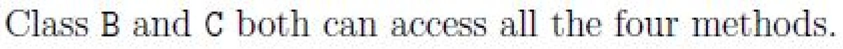
Options :

6406531958809. ✖ 

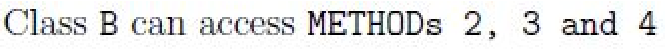
Class B can access METHODs 1, 2 and 4

6406531958810. ✖ 

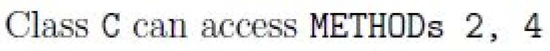
Class C can access METHODs 2, 3 and 4

6406531958811. ✖ 

Class B and C both can access all the four methods.

6406531958812. ✓ 

Class B can access `METHODs 2, 3 and 4`

6406531958813. ✓ 

Class C can access `METHODs 2, 4`

Sub-Section Number : 5

Sub-Section Id : 64065384391

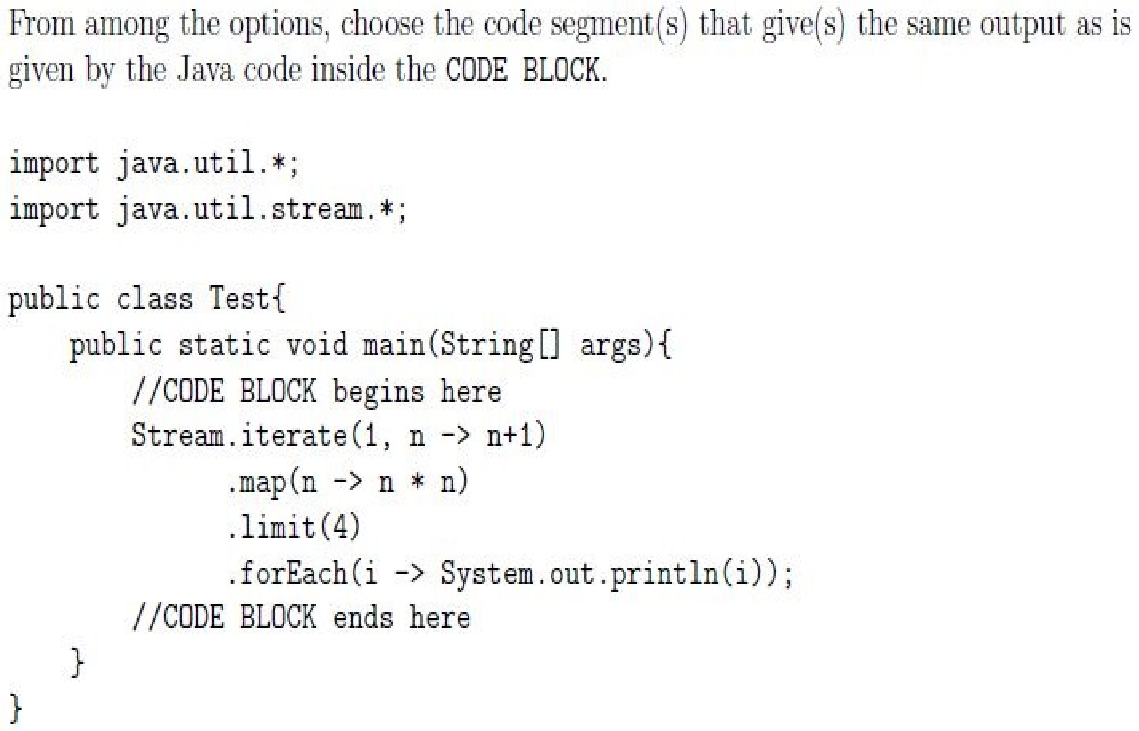
Question Shuffling Allowed : Yes

Is Section Default? : null

Question Number : 138 Question Id : 640653587039 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4 Max. Selectable Options : 0

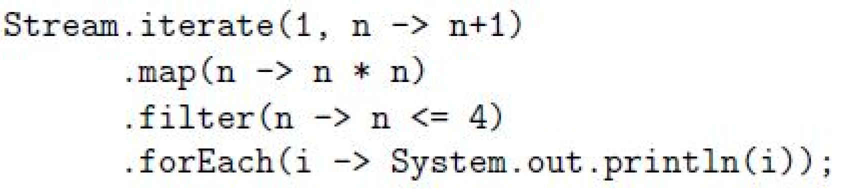
Question Label : Multiple Select Question



From among the options, choose the code segment(s) that give(s) the same output as is given by the Java code inside the `CODE BLOCK`.

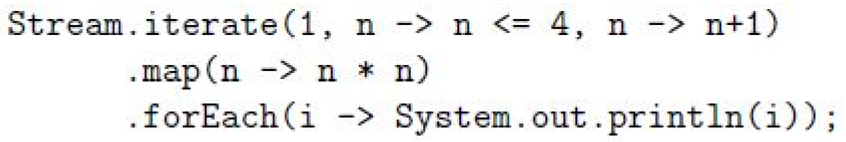
```
import java.util.*;
import java.util.stream.*;

public class Test{
    public static void main(String[] args){
        //CODE BLOCK begins here
        Stream.iterate(1, n -> n+1)
              .map(n -> n * n)
              .limit(4)
              .forEach(i -> System.out.println(i));
        //CODE BLOCK ends here
    }
}
```

Options :

6406531958796. ✖ 

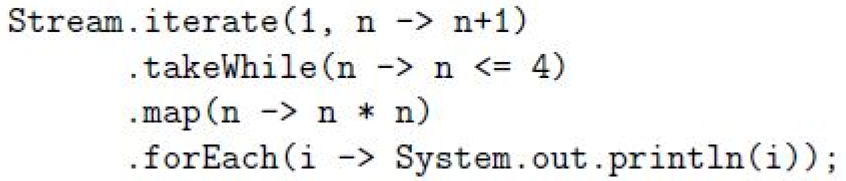
```
Stream.iterate(1, n -> n+1)
       .map(n -> n * n)
       .filter(n -> n <= 4)
       .forEach(i -> System.out.println(i));
```

6406531958797. ✓



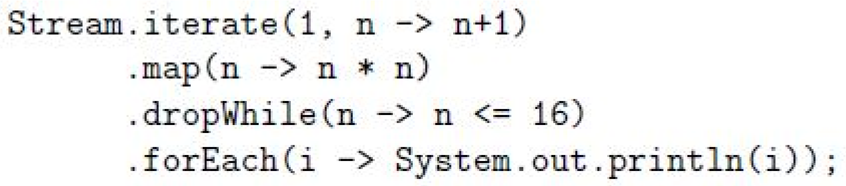
```
Stream.iterate(1, n -> n <= 4, n -> n+1)
      .map(n -> n * n)
      .forEach(i -> System.out.println(i));
```

6406531958798. ✓



```
Stream.iterate(1, n -> n+1)
      .takeWhile(n -> n <= 4)
      .map(n -> n * n)
      .forEach(i -> System.out.println(i));
```

6406531958799. ✖



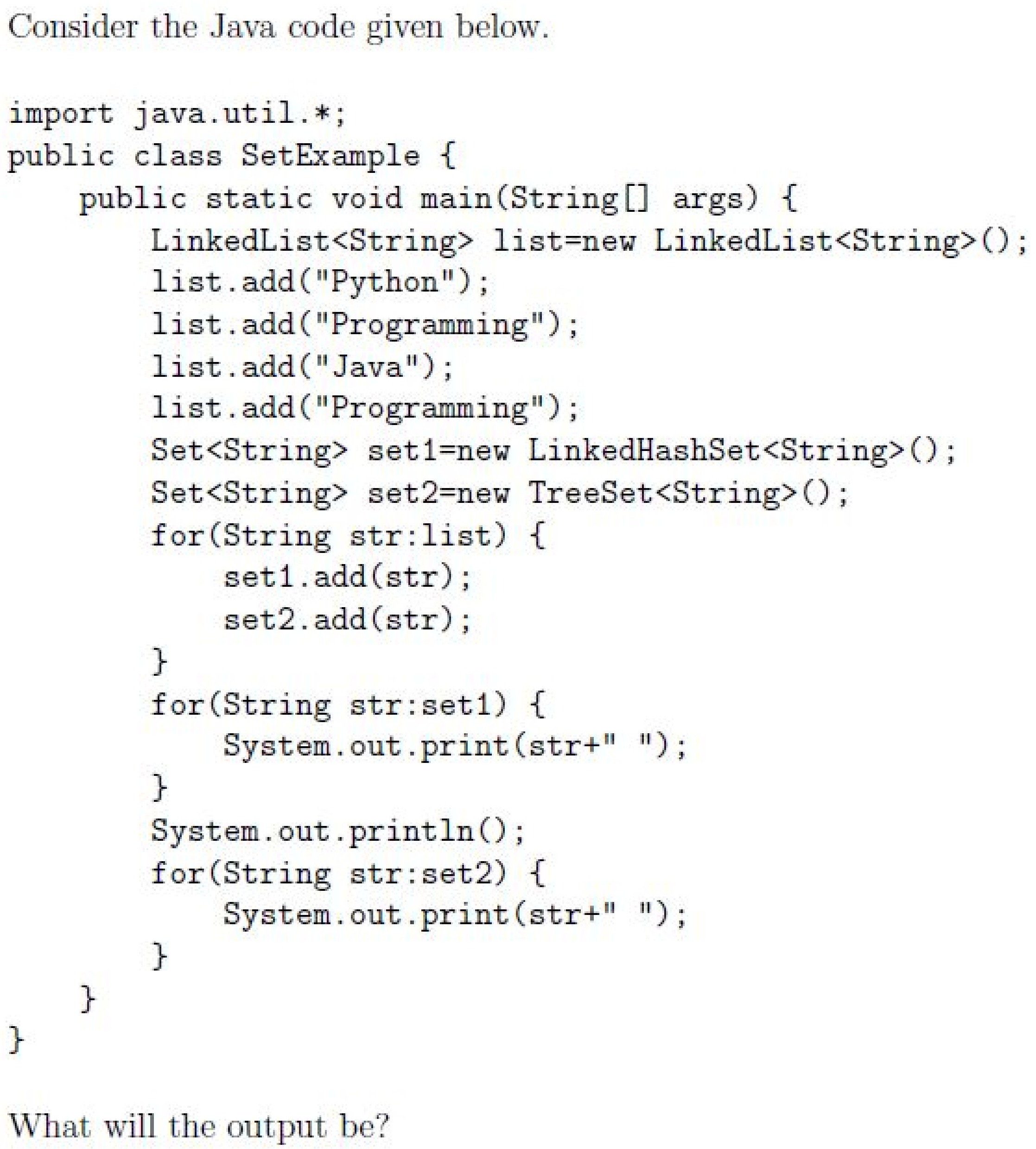
```
Stream.iterate(1, n -> n+1)
      .map(n -> n * n)
      .dropWhile(n -> n <= 16)
      .forEach(i -> System.out.println(i));
```

Sub-Section Number :	6
Sub-Section Id :	64065384392
Question Shuffling Allowed :	Yes
Is Section Default? :	null

Question Number : 139 Question Id : 640653587043 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 4

Question Label : Multiple Choice Question

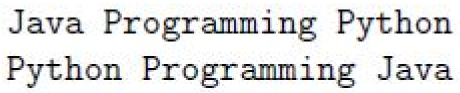


Consider the Java code given below.

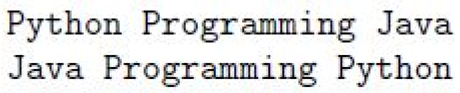
```
import java.util.*;
public class SetExample {
    public static void main(String[] args) {
        LinkedList<String> list=new LinkedList<String>();
        list.add("Python");
        list.add("Programming");
        list.add("Java");
        list.add("Programming");
        Set<String> set1=new LinkedHashSet<String>();
        Set<String> set2=new TreeSet<String>();
        for(String str:list) {
            set1.add(str);
            set2.add(str);
        }
        for(String str:set1) {
            System.out.print(str+" ");
        }
        System.out.println();
        for(String str:set2) {
            System.out.print(str+" ");
        }
    }
}
```

What will the output be?

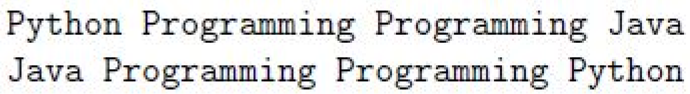
Options :

6406531958814. ✖ 

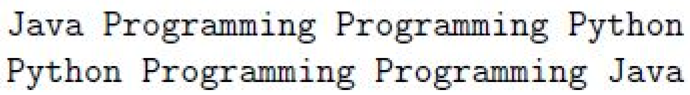
```
Java Programming Python
Python Programming Java
```

6406531958815. ✓ 

```
Python Programming Java
Java Programming Python
```

6406531958816. ✖ 

```
Python Programming Programming Java
Java Programming Programming Python
```

6406531958817. ✖ 

```
Java Programming Programming Python
Python Programming Programming Java
```

AppDev2

Section Id :	64065339716
Section Number :	10
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	17
Number of Questions to be attempted :	17
Section Marks :	50
Display Number Panel :	Yes
Group All Questions :	No
Enable Mark as Answered Mark for Review and Clear Response :	Yes
Maximum Instruction Time :	0
Sub-Section Number :	1
Sub-Section Id :	64065384393
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 140 Question Id : 640653587048 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0

Question Label : Multiple Choice Question

THIS IS QUESTION PAPER FOR THE SUBJECT "DIPLOMA LEVEL : MODERN APPLICATION DEVELOPMENT II (COMPUTER BASED EXAM) "

ARE YOU SURE YOU HAVE TO WRITE EXAM FOR THIS SUBJECT?