

Sub-Section Number :	8
Sub-Section Id :	640653112619
Question Shuffling Allowed :	Yes
Is Section Default? :	null

**Question Number : 155 Question Id : 640653770574 Question Type : SA Calculator : None**  
**Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**  
**Correct Marks : 3**

Question Label : Short Answer Question

The value of a function at point 5 is 10. The values of the function’s first and second order derivatives at this point are 5 and 2 respectively. What will be the function’s approximate value at the point 5.1? (Enter the answer correct up to two decimal places).

**Response Type :** Numeric  
**Evaluation Required For SA :** Yes  
**Show Word Count :** Yes

**Answers Type :** Range  
**Text Areas :** PlainText

**Possible Answers :**

10.2 to 10.8

## Java

Section Id :	64065353266
Section Number :	10
Section type :	Online
Mandatory or Optional :	Mandatory
Number of Questions :	16
Number of Questions to be attempted :	16
Section Marks :	100
Display Number Panel :	Yes

Section Negative Marks :	0
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 :	640653112620
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 156 Question Id : 640653770575 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 :

6406532577582.  YES

6406532577583.  NO

Sub-Section Number :	2
Sub-Section Id :	640653112621
Question Shuffling Allowed :	Yes
Is Section Default? :	null

**Question Number : 157 Question Id : 640653770576 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

**Question Label : Multiple Choice Question**

Consider the Java code given below.

The method `boolean containsKey (Object key)` in the class `Map` returns `true` if and only if the map contains a mapping for a key `k` such that `Objects.equals(key, k)`.

```
import java.util.*;
interface Shape {
    void draw();
}
class Circle implements Shape {
    public void draw() {
        System.out.println("Drawing a Circle");
    }
}
class Square implements Shape {
    public void draw() {
        System.out.println("Drawing a Square");
    }
}
class DrawingBoard<T extends Shape> {
    private Map<String, T> m = new HashMap<>(); // LINE 1
    public void add(String name, T shape) {
        m.put(name, shape);
    }
    public void draw(String name) {
        if (m.containsKey(name)) {
            T s = m.get(name); //LINE 2
            s.draw();
        } else {
            System.out.println("Shape not found");
        }
    }
}
public class Test {
    public static void main(String[] args) {
        DrawingBoard<Shape> dB = new DrawingBoard<Shape>();
        Shape s1 = new Circle();
        Shape s2 = new Square();
        dB.add("circle", s1);
        dB.add("square", s2);
        dB.draw("circle");
        dB.draw("triangle");
    }
}
```

Choose the correct option.

**Options :**

6406532577584. ✖ LINE 1 generates compilation error because type T is not known

This program generates output:

6406532577585. ✖ Drawing a Circle

LINE 2 generates compilation error because a variable of type Shape cannot refer to objects of type Circle and Square

6406532577586. ✖

This program generates output:

Drawing a Circle

Shape not found

6406532577587. ✔

**Question Number : 158 Question Id : 640653770582 Question Type : MCQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question

Consider two Java files located in two different packages as shown below.

A.java:

```
package com.pack1;
public class A {
    void methodOne() {
        System.out.println("Display methodOne");
    }
    private void methodTwo() {
        System.out.println("Display methodTwo");
    }
    protected void methodThree() {
        System.out.println("Display methodThree");
    }
    public void methodFour() {
        System.out.println("Display methodFour");
    }
}
```

B.java

```
package com.pack2;
import com.pack1.A;
public class B extends A {
    public static void main(String[] args) {
        B obj = new B();
        obj.methodOne();    //LINE 1
        obj.methodTwo();    //LINE 2
        obj.methodThree();  //LINE 3
        obj.methodFour();   //LINE 4
    }
}
```

Choose the correct option.

**Options :**

6406532577608. ✖ LINE 2 and LINE 3 generate compilation errors.

6406532577609. ✖ LINE 1, LINE 2, and LINE 3 generate compilation errors.

6406532577610. ✔ LINE 1 and LINE 2 generate compilation errors.

6406532577611. ✖ Only LINE 2 generates a compilation error.

**Question Number : 159 Question Id : 640653770583 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question

Consider the Java code given below.

```
class Validation {  
    public boolean validate(int a, int b) {  
        assert a > 0: "a should be greater than zero"; //LINE 1  
        assert b >= 1: "b should not be less than one"; //LINE 2  
        return true;  
    }  
}  
  
public class AssertTest {  
    public static void main(String[] args) {  
        int a = 1;  
        int b = -1;  
        int result = 0;  
        Validation obj = new Validation();  
        if (obj.validate(a, b))  
            result = a / b;  
        assert result > 0: "result should be greater than zero"; //LINE 3  
        System.out.println(result);  
    }  
}
```

Choose the correct option when the program is executed as:

java -ea AssertTest

**Options :**

6406532577612. ✖ LINE 1 generates assertion error.

6406532577613. ✔ LINE 2 generates assertion error.

6406532577614. ✖ LINE 3 generates assertion error.

6406532577615. ✖ This program does not generate assertion error.

**Question Number : 160 Question Id : 640653770585 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question



The following code maps a set of names of employees to their performance ratings, and groups the names based on their eligibility for bonus. Based on the code, answer the question that follows.

```
import java.util.*;
public class Employee {
    TreeSet<String> t1 = new TreeSet<String>();
    TreeSet<String> t2 = new TreeSet<String>();
    public boolean PerformanceRating(double rating) {
        if(rating >= 4.0){
            return true;
        }
        return false;
    }
    public void filterEmployees(TreeMap<String, Double> rating) {
        for (Map.Entry<String, Double> entry : rating.entrySet()) {
            if (PerformanceRating(entry.getValue())) {
                t1.add(entry.getKey());
            } else {
                t2.add(entry.getKey());
            }
        }
    }
    public void display() {
        System.out.println("Eligible for Bonus: " + t1);
        System.out.println("Not Eligible for Bonus: " + t2);
    }
    public static void main(String[] args) {
        TreeMap<String, Double> rating = new TreeMap<String, Double>();
        rating.put("Ramesh", 4.5);
        rating.put("Suresh", 3.8);
        rating.put("Kartik", 4.2);
        rating.put("Shubham", 3.5);
        rating.put("Mukesh", 4.8);
        Employee e = new Employee();
        e.filterEmployees(rating);
        e.display();
    }
}
```

Choose the correct option.

**Options :**

This program generates the output:

Eligible for Bonus: [Shubham, Suresh]

Not Eligible for Bonus: [Mukesh, Ramesh, Kartik]

6406532577620. ✖

6406532577621. ✔



This program generates the output:

Eligible for Bonus: [Kartik, Mukesh, Ramesh]

Not Eligible for Bonus: [Shubham, Suresh]

This program generates the output:

Eligible for Bonus: [Mukesh, Ramesh, Kartik]

6406532577622. ✖ Not Eligible for Bonus: [Shubham]

6406532577623. ✖ The order in which elements of t1 and t2 are printed cannot be predicted.

**Question Number : 161 Question Id : 640653770586 Question Type : MCQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question

Consider the following code.

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        List<Integer> scores1 = new ArrayList<>();
        scores1.add(34);
        scores1.add(42);
        scores1.add(50);

        List<Integer> scores2 = new ArrayList<>();
        scores2.add(45);
        scores2.add(90);
        scores2.add(34);

        Map<String, Integer> am = new HashMap<>();
        Map<String, List<Integer>> hm = new HashMap<>();
        hm.put("Anil", scores1);
        hm.put("Vikas", scores2);
        Set<String> names = hm.keySet();
        for(String name : names){
            List<Integer> temp = hm.get(name);
            int count = 0;
            int sum = 0;
            ***-----***
                CODE BLOCK
            ***-----***
            int avg = sum/count;
            am.put(name, avg);
        }
    }
}
```

Choose the correct option to fill in the CODE BLOCK to add the name and the average scores of both the students as map entries in Map<String, Integer> am.

**Options :**

```
        for(List i : temp){
            count = count + 1;
            sum = sum + i;
        }
```

6406532577624. ✖

6406532577625. ✖

```
for(List<Integer> i : temp){  
    count = count + 1;  
    sum = sum + i;  
}
```

```
        for(int i: temp){  
            sum = sum + temp;  
            count = count + 1;  
6406532577626. ✖ }  
}
```

```
        for(Integer i : temp){  
            count = count + 1;  
            sum = sum + i;  
6406532577627. ✔ }  
}
```

**Question Number : 162 Question Id : 640653770587 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question

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

```
import java.util.*;
public class PalindromeChecker {
    public static boolean isPalindrome(String word) {
        Deque<Character> deque = new ArrayDeque<>();
        for (int i = 0; i < word.length(); i++) {
            deque.add(word.charAt(i));
        }
        //CODE BLOCK
        return true;
    }
    public static void main(String[] args) {
        String word1 = "radar";
        String word2 = "hello";
        System.out.println("Is '" + word1 + "' a palindrome? " +
                           isPalindrome(word1));
        System.out.println("Is '" + word2 + "' a palindrome? " +
                           isPalindrome(word2));
    }
}
```

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

Is 'radar' a palindrome? true

Is 'hello' a palindrome? false

Please note the following methods from type Deque.

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

pollFirst(): Retrieves and removes the first element of the deque, or returns null if the deque is empty.

**Options :**

```
while (deque.size() > 0) {
    if (deque.pollFirst() != deque.pollLast()) {
        return true;
    }
}
```

6406532577628. ✖

6406532577629. ✖

```
while (deque.size() < 0) {  
    if (deque.pollFirst() != deque.pollLast()) {  
        return false;  
    }  
}
```

```
        while (deque.size() > 1) {  
            if (deque.pollFirst() != deque.pollLast()) {  
                return false;  
            }  
        }  
6406532577630. ✓
```

```
        while (deque.size() > 0) {  
            char first = deque.pollFirst();  
            char last = deque.pollLast();  
            if (first != last) {  
                return true;  
            }  
        }  
6406532577631. ✖
```

**Question Number : 163 Question Id : 640653770588 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question



Consider the Java code given below.

```
class Student implements Cloneable {
    String studentName;
    public Student(String n) {
        studentName = n;
    }
    public Student clone() throws CloneNotSupportedException {
        return (Student) super.clone();
    }
}
class Game implements Cloneable {
    String gameName;
    Student student1;
    Student student2;
    public Game(String gN, Student s1, Student s2) {
        gameName = gN;
        student1 = s1;
        student2 = s2;
    }
    public Game clone() throws CloneNotSupportedException {
        Game g = (Game) super.clone();
        g.student1 = g.student1.clone();
        g.student2 = g.student2.clone();
        return g;
    }
}
public class Test {
    public static void main(String[] args) throws CloneNotSupportedException {
        Student s1 = new Student("Ramesh");
        Student s2 = new Student("Jogesh");
        Game obj1 = new Game("ABC", s1, s2);
        Game obj2 = obj1.clone();
        obj2.student1.studentName = "Shubham";
        obj2.gameName = "XYZ";
        System.out.println(obj1.gameName + " : " + obj1.student1.studentName);
        System.out.println(obj2.gameName + " : " + obj2.student1.studentName);
    }
}
```

What will the output be?

**Options :**

ABC : Shubham

ABC : Shubham

6406532577632. ✖

XYZ : Shubham

XYZ : Shubham

6406532577633. ✖



ABC : Ramesh  
6406532577634. ✓ XYZ : Shubham

ABC : Shubham  
6406532577635. ✗ XYZ : Shubham

**Question Number : 164 Question Id : 640653770589 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 6**

Question Label : Multiple Choice Question

Consider the Java code given below.

```
import java.util.*;
import java.util.stream.*;
public class MyClass {
    public static void main(String args[]) {
        List<String> wordlist = new ArrayList<>();
        wordlist.add("reluctant");
        wordlist.add("test");
        wordlist.add("unpleasant");
        wordlist.add("delicious");
        wordlist.add("away");
        Stream<String> startLongWords = wordlist.stream()
            .filter(w -> w.length() > 5)
            .map(s -> s.substring(0, 2));
        startLongWords.forEach(System.out::println);
    }
}
```

What will the output be?

**Options :**

6406532577636. ✗

re  
te  
un  
de  
aw

6406532577637. ✖ te  
aw

6406532577638. ✔ re  
un  
de

6406532577639. ✖ reluctant  
unpleasant  
delicious

Sub-Section Number : 3  
Sub-Section Id : 640653112622  
Question Shuffling Allowed : Yes  
Is Section Default? : null

Question Number : 165 Question Id : 640653770578 Question Type : MCQ Is Question  
Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction  
Time : 0  
Correct Marks : 7  
Question Label : Multiple Choice Question

Consider the Java code given below.

```
interface Chargeable {
    void charge();
}
class SmartPhone implements Chargeable {
    public void charge() {
        System.out.println("Charging SmartPhone");
    }
}
class SmartWatch implements Chargeable {
    public void charge() {
        System.out.println("Charging SmartWatch");
    }
}
class DeviceList {
    private Object[] cArr = {new SmartPhone(), new SmartWatch()};
    public void chargeDevices() {
        for (int i = 0; i < cArr.length; i++) {
            // LINE 1
        }
    }
}
public class Test {
    public static void main(String[] args) {
        DeviceList dList = new DeviceList();
        dList.chargeDevices();
    }
}
```

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

Charging SmartPhone

Charging SmartWatch

**Options :**

6406532577592. ✖ cArr[i].charge();

6406532577593. ✔ ((Chargeable)cArr[i]).charge();

6406532577594. ✖ ((SmartPhone)cArr[i]).charge();

6406532577595. ✖ ((SmartWatch)cArr[i]).charge();

**Question Number : 166 Question Id : 640653770579 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 7**

Question Label : Multiple Choice Question

Consider the Java code given below.

```
public class ArrayUtils{
    public <T extends Comparable> T max(T[] arr){
        // code for finding maximum here
    }
    public <T extends Number> T avg(T[] arr){
        // code for finding average of elements here
    }
    public <T> int count(T[] arr){
        // code for counting the number of elements in array
    }
}
```

How does class ArrayUtils look after type erasure?

**Options :**

```
public class ArrayUtils{
    public Object max(Object[] arr){
        // code for finding maximum here
    }
    public Number avg(Number[] arr){
        // code for finding average of elements here
    }
    public int count(Object[] arr){
        // code for counting the number of elements in array
    }
}
```

6406532577596. ✖

6406532577597. ✖

```

public class ArrayUtils{
    public Object max(Object[] arr){
        // code for finding maximum here
    }
    public Object avg(Object[] arr){
        // code for finding average of elements here
    }
    public int count(Object[] arr){
        // code for counting the number of elements in array
    }
}

```

```

public class ArrayUtils{
    public Comparable max(Comparable[] arr){
        // code for finding maximum here
    }
    public Number avg(Number[] arr){
        // code for finding average of elements here
    }
    public int count(Object[] arr){
        // code for counting the number of elements in array
    }
}

```

6406532577598. ✓

```

public class ArrayUtils{
    public Comparable max(Comparable[] arr){
        // code for finding maximum here
    }
    public Integer avg(Integer[] arr){
        // code for finding average of elements here
    }
    public int count(Object[] arr){
        // code for counting the number of elements in array
    }
}

```

6406532577599. ✖

Question Number : 167 Question Id : 640653770581 Question Type : MCQ Is Question

Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction



**Time : 0**

**Correct Marks : 7**

Question Label : Multiple Choice Question

Consider the Java code given below.

```
class MinimumMarksException extends Exception {
    public MinimumMarksException(String message) {
        super(message);
    }
}
class Student {
    private double marks;
    private final double MINIMUM_MARKS = 40.0;
    // Constructor to initialize the marks
    public void checkResult() throws MinimumMarksException {
        if (marks < MINIMUM_MARKS) {
            throw new MinimumMarksException("Minimum marks not scored");
        }
        else {
            System.out.println("Scored sufficient marks");
        }
    }
}
public class Test {
    public static void main(String[] args) {
        Student s1 = new Student(55.0);
        Student s2 = new Student(30.0);
        try {
            s1.checkResult();
            s2.checkResult();
        } catch (MinimumMarksException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Choose the correct option.

**Options :**

This program generates the output:

Scored sufficient marks

6406532577604. ✓ Error: Minimum marks not scored



6406532577605. ✖ This program generates the output:  
Scored sufficient marks  
MinimumMarksException

6406532577606. ✖ This program generates the output:  
Scored sufficient marks  
Minimum marks not scored

6406532577607. ✖ This program generates the output:  
Error: MinimumMarksException

**Question Number : 168 Question Id : 640653770584 Question Type : MCQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**  
**Correct Marks : 7**  
Question Label : Multiple Choice Question

Consider the Java code given below.

```
import java.util.*;
class Contestant{
    String name;
    int points;
    //constructor to initialize name and points
    public String toString() {
        return name;
    }
}
public class IteratorTest {
    public static boolean ranked(int x) {
        if(x < 5)
            return false;
        return true;
    }
    public static void getFinalList(List<Contestant> cList){
        Iterator<Contestant> it = cList.iterator();
        while (it.hasNext()) {
            Contestant c = it.next();
            if(!ranked(c.points))
                ----- //LINE 1
        }
    }
    public static void main(String[] args) {
        var list = new ArrayList<Contestant>();
        list.add(new Contestant("Sanju", 7));
        list.add(new Contestant("Kiran", 4));
        list.add(new Contestant("Ram", 5));
        list.add(new Contestant("John", 5));
        getFinalList(list);
        System.out.println(list);
    }
}
```

Choose the correct option to be filled in place of LINE 1 so that the output is:  
[Sanju, Ram, John]

**Options :**

6406532577616. ✖ it.remove(c)

6406532577617. ✔ it.remove()

6406532577618.

✖ cList.remove()

6406532577619. ✖ cList.remove(c)

<b>Sub-Section Number :</b>	4
<b>Sub-Section Id :</b>	640653112623
<b>Question Shuffling Allowed :</b>	Yes
<b>Is Section Default? :</b>	null

**Question Number : 169 Question Id : 640653770577 Question Type : MSQ Is Question Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 8 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the Java code given below.

```
public interface Animal{
    public abstract void makeSound();
}
public class Zoo{
    private String category;
    public void setCategory(String s) {
        this.category = s;
    }
    public String getCategory() {
        return category;
    }
    public Animal createAnimal(){
        if(getCategory() == "Mammal"){
            return new Mammal();
        }
        return new Bird();
    }
    private class Mammal implements Animal{
        public void makeSound(){
            System.out.println("Mammal making a sound");
        }
    }
    private class Bird implements Animal {
        public void makeSound() {
            System.out.println("Bird chirping");
        }
    }
}
public class Example {
    public static void main(String[] args) {
        Zoo z = new Zoo();
        z.setCategory("Mammal");
        // ----- Line 1 -----
    }
}
```

Identify the appropriate option to fill in place of LINE 1 such that the output is:  
Mammal making a sound

**Options :**

6406532577588. ✖ z.makeSound();

6406532577589. ✔ z.createAnimal().makeSound();

6406532577590.

✓ ((Animal) z.createAnimal()).makeSound();

```
Animal a = z.createAnimal();  
6406532577591. ✓ a.makeSound();
```

**Question Number : 170 Question Id : 640653770580 Question Type : MSQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 8 Max. Selectable Options : 0**

Question Label : Multiple Select Question

Consider the Java code given below that prints the highest goals among a set of given GoalScorer objects. From among the options, identify the appropriate function header for the function printHighestGoals that takes as input an array of GoalScorer objects and prints the highest goal.

```
import java.util.*;  
interface GoalScorer {  
    public abstract int getGoals();  
}  
public class Player implements GoalScorer {  
    private double goals;  
    // Constructor  
    // method getGoals() that returns goals  
}  
public class Test {  
    // LINE 1: FUNCTION HEADER  
    {  
        // invokes method getGoals()  
        // to print the value of highest goals  
    }  
    public static void main(String[] args) {  
        GoalScorer[] players = {new Player(123), new Player(98), new Player(79)};  
        printHighestGoals(players);  
    }  
}
```

Choose the correct option(s).

**Options :**

6406532577600. ✖ `public static void printHighestGoals(<?> players)`

6406532577601. ✔ `public static <T extends GoalScorer> void printHighestGoals(T[] players)`

6406532577602. ✖ `public static <T extends Player> void printHighestGoals(T[] players)`

6406532577603. ✔ `public static void printHighestGoals(GoalScorer[] players)`

**Question Number : 171 Question Id : 640653770590 Question Type : MSQ Is Question**

**Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0**

**Correct Marks : 8 Max. Selectable Options : 0**

Question Label : Multiple Select Question



Consider the Java code given below that should print the names of students whose gpa is between 3.0 and 3.8 (both inclusive).

```
import java.util.*;
class Student {
    String name;
    double gpa;
    public Student(String n, double g) {
        name = n;
        gpa = g;
    }
}
public class StreamExample {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student("Alice", 3.5));
        students.add(new Student("Bob", 3.2));
        students.add(new Student("Charlie", 3.8));
        students.add(new Student("David", 3.0));
        students.add(new Student("Eva", 4.0));
        //CODE BLOCK
    }
}
```

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

### Options :

```
students.stream()
    .map(s -> s.gpa >= 3.0 && s.gpa <= 3.8)
    .forEach(s -> System.out.println(s.name));
```

6406532577640. ✖

```
students.stream()
    .filter(s -> s.gpa >= 3.0 && s.gpa <= 3.8)
    .forEach(s -> System.out.println(s.name));
```

6406532577641. ✔

```
students.stream()
    .filter(s -> s.gpa >= 3.0)
    .filter(s -> s.gpa <= 3.8)
    .forEach(s -> System.out.println(s.name));
```

6406532577642. ✔

6406532577643. ✖

```
students.stream()  
    .filter(s -> s.gpa >= 3.0)  
    .map(s -> s.gpa <= 3.8)  
    .forEach(s -> System.out.println(s.name));
```

## AppDev2

Section Id :	64065353267
Section Number :	11
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
Section Negative Marks :	0
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 :	640653112624
Question Shuffling Allowed :	No
Is Section Default? :	null

Question Number : 172 Question Id : 640653770591 Question Type : MCQ Is Question

Mandatory : No Calculator : None Response Time : N.A Think Time : N.A Minimum Instruction Time : 0

Correct Marks : 0