BSCCS2005: ET Set 1

1. Consider the Java code given below. [MCQ: 4 marks]

```
class Monitor {
    public void screenSize() {
        System.out.println("Normal screen size");
    }
    public void resolution() {
        System.out.println("Normal resolution");
    }
}
class LCD extends Monitor {
    public void screenSize() {
        System.out.println("Screen size is large");
    }
}
class LED extends Monitor {
    public void screenSize() {
        System.out.println("Screen size is medium");
    }
    public void resolution() {
        System.out.println("HD resolution");
    }
}
public class Test {
    static void show(Monitor[] monitors) {
        for (int i = 0; i < monitors.length; i++) {
            monitors[i].screenSize();
            monitors[i].resolution();
        }
    }
    public static void main(String[] args) {
        Monitor[] monitors = {new LCD(), new LED()}; // LINE 1
        show(monitors);
    }
}
```

Choose the correct option.

◯ This code generates the output:

```
Screen size is medium
HD resolution
Screen size is medium
HD resolution
```

◯ This code generates the output:

```
Normal screen size
Normal resolution
Normal screen size
Normal resolution
```

√ This code generates the output:

```
Screen size is large
Normal resolution
Screen size is medium
HD resolution
```

◯ Compilation error at `LINE 1` because a reference variable of type `Monitor` cannot refer to an object of class `LED`

2. Consider the code given below that checks whether two candidates are from the same college. Method `equals` is overridden to compare two `Candidate` objects as follows. If two candidates are from the same college then they are said to be equal. Based on the given information, answer the question that follows. [MCQ: 4 marks]

```
class Candidate {
    private String name;
    private String college;
    // Constructor to initialize instance variables
    public String toString() {
        return name;
    }

    public boolean equals(Object obj) {
        // CODE BLOCK
    }
}
public class Test {
    public static void main(String[] args) {
        Candidate c1 = new Candidate("Shreya", "IITMadras");
        Candidate c2 = new Candidate("Hari", "IITDelhi");
        Candidate c3 = new Candidate("Aisha", "IITMadras");
        if (c1.equals(c3)) {
            System.out.println(c1 + " and " + c3 + " belong to the same college");
        }
        if (c2.equals(c3)) {
            System.out.println(c2 + " and " + c3 + " belong to the same college");
        }
    }
}
```

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

```
Shreya and Hari belong to the same college
```

◯
```
if(obj instanceof Candidate) {
   if(this.college.equals(obj.college))
       return true;
   }
   return false;
```

◯
```
if(this.college.equals(obj.college))
    return true;
return false;
```

√
```
if(obj instanceof Candidate) {
    Candidate c = (Candidate) obj;
    if(this.college.equals(c.college))
        return true;
}
return false;
```

◯
```
if(obj instanceof Candidate) {
     Candidate c = obj;
     if(this.college.equals(c.college))
         return true;
}
return false;
```

3. Consider the Java code given below. [Copy constructor : MCQ: 4 marks]

```
class Intern {
    private String name;
    public Intern(String n) {
        name = n;
    }
    public Intern(Intern i) {
        this.name = i.name;
    }
    public void setName(String n) {
        name = n;
    }
    public String getName() {
        return name;
    }
}
public class Test {
    public static void main(String[] args) {
        Intern i1 = new Intern("Jaya");
        Intern i2 = new Intern(i1);
        Intern i3 = i1;
        i1.setName("Subash");
        System.out.println(i1.getName());
        System.out.println(i2.getName());
        System.out.println(i3.getName());
    }
}
```

What will the output be?

◯
```
Subash
Jaya
Jaya
```

√
```
Subash
Jaya
Subash
```

◯
```
Subash
Subash
Subash
```

◯
```
Subash
Subash
Jaya
```

4. Consider the code given below. [MCQ: 4 marks]

```
class Device {
    public void powerOn() {
        System.out.println("Device is on");
    }
}
class Mobile extends Device {
    public void display() {
        System.out.println("Mobile display");
    }
class Smartphone extends Mobile {
    public void display() {
        System.out.println("Smartphone display");
    }
    public void connect() {
        System.out.println("Connected to Internet");
    }
}

public class TestDevice {
    public static void main(String[] args) {
        Device d = new Mobile();
        Mobile m = new Smartphone(); // LINE 1
        d.powerOn();
        ((Mobile)d).display(); // LINE 2
        m.connect();  // LINE 3
    }
}
```

Choose the correct option.

- ◯ `LINE 1` generates a compilation error because a variable of type `Mobile` cannot refer to an object of type `Smartphone`.
- ◯ `LINE 2` generates a compilation error because a variable of type `Device` cannot be type cast to an object of type `Mobile`.
- √ `LINE 3` generates a compilation error because the method `connect()` is not defined in class `Mobile`.
- ◯ This code generates the output:

  ```
  Device is on
  Mobile display
  Connected to Internet
  ```

5. Consider the Java code given below. [MCQ: 4 marks]

```
import java.util.*;
public class Test{
    public static void main(String[] args) {
        ArrayDeque<String> queue1 = new ArrayDeque<String>();
        queue1.add("Violet");
        queue1.addFirst("Yellow");
        queue1.add("Pink");
        queue1.addFirst("Blue");
        queue1.add("Blue");
        System.out.println(queue1);
        TreeSet<String> set = new TreeSet<String>(queue1);
        System.out.println(set);
    }
}
```

What will the output be?

√ This program generates the output:
```
[Blue, Yellow, Violet, Pink, Blue]
[Blue, Pink, Violet, Yellow]
```

◯ This program generates the output:
```
[Blue, Pink, Violet, Yellow]
[Blue, Yellow, Violet, Pink, Blue]
```

◯ This program generates the output:
```
[Blue, Pink, Violet, Yellow]
[Blue, Pink, Violet, Yellow]
```

◯ This program generates the output:
```
[Blue, Yellow, Violet, Pink, Blue]
[Blue, Yellow, Violet, Pink, Blue]
```

6. Consider the Java code given below. [MCQ: 4 marks]

```
class Chef {
    String name;
    public Chef(String n) {
        name = n;
    }
}
class Dish implements Cloneable {
    String dishName;
    Chef[] chefs;
    public Dish(String name, Chef[] chefs) {
        dishName = name;
        this.chefs = chefs;
    }
    public Dish clone() throws CloneNotSupportedException {
        Dish d = (Dish) super.clone();
        d.chefs = this.chefs.clone();
        return d;
    }
}
public class Test {
    public static void main(String[] args) throws CloneNotSupportedException {
        Chef[] chefs1 = { new Chef("Ravi"), new Chef("Raju") };
        Dish d1 = new Dish("Biriyani", chefs1);
        Dish d2 = d1.clone();
        Chef[] chefs2 = d2.chefs;
        chefs2[0].name = "Veena";
        d2.dishName = "FriedRice";

        System.out.println(d1.dishName + " : " + d1.chefs[0].name);
        System.out.println(d2.dishName + " : " + d2.chefs[0].name);
    }
}
```

What will the output be?

- ◯ `Biriyani : Veena`
  `Biriyani : Veena`
- ◯ `FriedRice : Veena`
  `FriedRice : Veena`
- ◯ `Biriyani : Ravi`
  `FriedRice : Veena`

√ Biriyani : Veena
  FriedRice : Veena

7. Consider the Java code given below. [MCQ: 4 marks]

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Date");
        list.add("Durian");
        list.add("Banana");
        list.add("Cherry");
        list.add("Dragonfruit");

        list.stream().takeWhile(s -> s.startsWith("D"))
            .forEach(s -> System.out.print(s + " "));

        System.out.println();

        list.stream().dropWhile(s -> s.startsWith("D"))
            .forEach(s -> System.out.print(s + " "));

    }
}
```

What will the output be?

◯ Date Durian Dragonfruit
Banana Cherry

◯ Date Durian
Banana Cherry

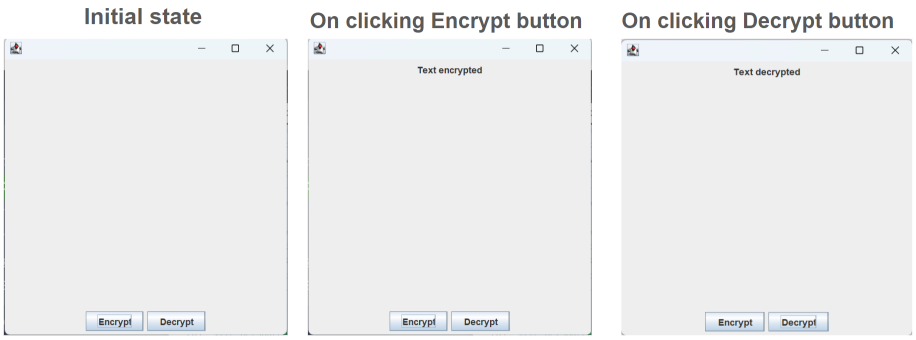
√ Date Durian
Banana Cherry Dragonfruit

◯ Date Durian Banana Cherry Dragonfruit

8. Consider the Java code given below. [MSQ: 6 marks]

```
import javax.swing.*;
import java.awt.event.*;
public class ButtonEventTest extends JFrame implements ActionListener{
    private JButton b1, b2;
    private JLabel l1;
    JPanel panel1, panel2;
    public ButtonEventTest() {
        b1 = new JButton("Encrypt");
        b2 = new JButton("Decrypt");
        panel1 = new JPanel();
        panel1.add(b1);
        panel1.add(b2);
        add(panel1, "South");

        l1 = new JLabel("");
        panel2 = new JPanel();
        panel2.add(l1);
        add(panel2, "North");

        setVisible(true);
        setSize(400, 400);

        b1.setActionCommand("action1");
        b2.setActionCommand("action2");

        b1.addActionListener(this);
        b2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        //CODE SEGMENT
    }
    public static void main(String[] args) {
        new ButtonEventTest();
    }
}
```

Choose the correct code segment(s) to be filled inside method `actionPerformed()` such that on clicking the `Encypt` button, the label text changes to `Text encrypted` and on clicking the `Decrypt` button, the label text changes to `Text decrypted`.



√
```
if(e.getActionCommand().equals("action1"))
    l1.setText("Text encrypted");
else if(e.getActionCommand().equals("action2"))
    l1.setText("Text decrypted");
```

◯
```
if(e.getActionCommand().equals("b1"))
    l1.setText("Text encrypted");
else if(e.getActionCommand().equals("b2"))
    l1.setText("Text decrypted");
```

√
```
if(e.getSource().equals(b1))
    l1.setText("Text encrypted");
else if(e.getSource().equals(b2))
    l1.setText("Text decrypted");
```

◯
```
if(e.getSource().equals("action1"))
    l1.setText("Text encrypted");
else if(e.getSource().equals("action2"))
    l1.setText("Text decrypted");
```

9. Consider the Java code given below. [MCQ: 5 marks]

```
import javax.swing.*;
import java.awt.*;
public class GUITest extends JFrame {
    JPanel pnlLbl, pnlTxt, pnlBtn;
    JLabel lblId, lblPwd;
    JTextField txtId, txtPwd;
    JButton btn;
    public GUITest() {
        lblId = new JLabel("Phone no:");
        lblPwd = new JLabel("OTP:");
        txtId = new JTextField(10);
        txtPwd = new JTextField(10);
        btn = new JButton("Login");
        pnlLbl = new JPanel();
        //add lblId and txtId to pnlLbl
        pnlTxt = new JPanel();
        //add lblPwd and txtPwd to pnlTxt
        pnlBtn = new JPanel();
        //add btn to pnlBtn

        //CODE BLOCK

        setVisible(true);
        setSize(300,200);
    }
    public static void main(String[] args) {
        new GUITest();
    }
}
```

Choose the correct option to be filled in place of `CODE BLOCK` such that the above program produces the GUI given below.

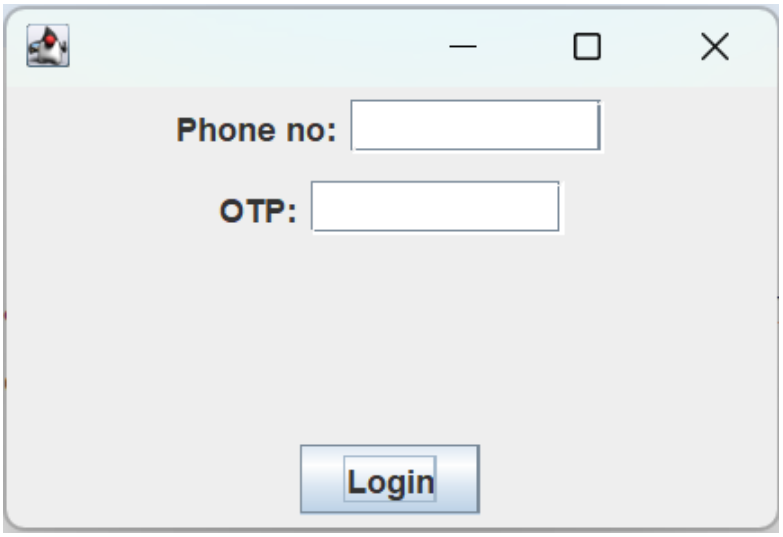


Figure 1

◯
```
add(pnlLbl, "Center");
add(pnlTxt, "North");
add(pnlBtn, "South");
```

◯
```
add(pnlLbl, "North");
add(pnlTxt, "South");
add(pnlBtn, "Center");
```

√
```
add(pnlLbl, "North");
add(pnlTxt, "Center");
add(pnlBtn, "South");
```

◯
```
add(pnlLbl, "South");
add(pnlTxt, "Center");
add(pnlBtn, "North");
```

10. Consider the Java code given below. [MCQ: 4 Marks]

```
1 class ClassOne{
2      public void methodOne(){
3          // ...
4          methodTwo();
5          // ...
6       }
7       public void methodTwo(){
8          // ...
9       }
10 }
11 class ClassTwo{
12     public static void methodThree(){
13         // ...
14         ClassOne c = new ClassOne();
15         c.methodOne();
16         // ...
17     }
18     public static void methodFour(){
19         // ...
20         methodThree();
21         // ...
22     }
23     public static void main(String[] args) {
24         // ...
25         methodFour();
26     }
27 }
```

During the execution of Line 16 in the above code, the activation record of which method is at the top of the stack of activation records?

- ◯ `main`
- ◯ `methodOne`
- ◯ `methodTwo`
- √ `methodThree`
- ◯ `methodFour`

11. Consider the Java code given below. [MCQ : 4 Marks]

```
interface TransportService {
    void bookRide();
}
class TransportApp {
    public TaxiService getTaxiService() {
        return new TaxiService();
    }
    public BusService getBusService() {
        return new BusService();
    }
    private class TaxiService implements TransportService {
        public void bookRide() {
            System.out.println("Booking a taxi");
        }
    }
    private class BusService implements TransportService {
        public void bookRide() {
            System.out.println("Booking a bus");
        }
    }
}
public class Test {
    public static void main(String[] args) {
        TransportApp t = new TransportApp();
        //CODE BLOCK
        obj1.bookRide();
        obj2.bookRide();
    }
}
```

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

```
Booking a taxi
Booking a bus
```

◯
```
TaxiService obj1 = new TaxiService();
BusService obj2 = new BusService();
```

√
```
TransportService obj1 = t.getTaxiService();
TransportService obj2 = t.getBusService();
```

◯
```
TaxiService  obj1 = t.getTaxiService();
BusService obj2 = t.getBusService();
```

◯
```
TransportService obj1 = new TaxiService();
TransportService obj2 = new BusService();
```

12. Consider the Java code given below. [MCQ : 4 Marks]

```
interface OnlineCourse {
    default void showDetails() {
        System.out.println("Course duration is 3 months");
    }
    default void enroll() {
        System.out.println("Enrolled");
    }
}
class CloudCourse implements OnlineCourse { //LINE 1
    public void enroll() {
        System.out.println("Enrolled in cloud course");
    }
}
class MLCourse implements OnlineCourse {  //LINE 2
    public void showDetails() {
        System.out.println("ML Course");
    }
}
public class Test {
    public static void main(String[] args) {
        OnlineCourse courses[] = new OnlineCourse[2];
        courses[0] = new CloudCourse();
        courses[1] = new MLCourse();
        for (OnlineCourse course : courses) {
            course.showDetails();
            course.enroll();
        }
    }
}
```

Choose the correct option.

◯ Compilation error at `LINE 1` because method `showDetails()` is not overridden in class `CloudCourse`

◯ Compilation error at `LINE 2` because method `enroll()` is not overridden in class `MLCourse`

◯ This program generates the output:

```
Enrolled in cloud course
ML Course
```

√ This program generates the output:

```
Course duration is 3 months
Enrolled in cloud course
MLCourse
Enrolled
```

13. Consider the Java code given below. [MCQ : 4 Marks]

```
abstract class Bag {
    abstract void open();
    void carry() {       // LINE 1
        System.out.println("Carrying bag");
    }
}
class Backpack extends Bag {
    void open() {
        System.out.println("Opening backpack");
    }
    void carry() {
        System.out.println("Wearing backpack");
    }
}
class ToteBag extends Bag {
    void open() {
        System.out.println("Opening tote bag");
    }
    void carry() {
        System.out.println("Holding tote bag");
    }
}
public class Test {
    public static void main(String[] args) {
        Bag bag1 = new Backpack(); // LINE 2
        Bag bag2 = new ToteBag();  // LINE 3
        bag1.carry();
        bag1.open();
        bag2.carry();
        bag2.open();
    }
}
```

Choose the correct option.

◯ `LINE 1` generates compilation error because abstract class must contain only abstract methods.

◯ `LINE 2` and `LINE 3` generate compilation errors because reference variable of type `Bag` cannot store the objects of type `Backpack` and `ToteBag`.

√ This program generates the output:

```
Wearing backpack
Opening backpack
```

```
Holding tote bag
Opening tote bag
```

◯ This program generates the output:

```
Carrying bag
Opening backpack
Carrying bag
Opening tote bag
```

14. Consider the Java code given below. [MCQ : 4 Marks]

```
interface Iterator{
    public boolean has_next();
    public Object get_next();
}
abstract class Printable{
    public abstract void print();
}
class ProductList{
    private final int limit = 3;
    private Product[] list = { new Product("Laptop", "P1001"),
            new Product("Smartphone", "P1002"),
            new Product("Smartwatch", "P1003")
    };
    private class Product extends Printable{
        private String name, productId;
        //Constructor to initialize instance variables
        public void print() {
            System.out.println(productId + ", " + name);
        }
    }
    private class ProdIter implements Iterator{
        private int indx;
        public ProdIter() {
            //constructor
        }
        public boolean has_next() {
            //if next element available in list return true;
            //else false
        }
        public Object get_next() {
            //return next element from list
        }
    }
    public Iterator getIterator() {
        return new ProdIter();
    }
}
public class IterTest {
    public static void main(String[] args) {
        ProdList pList = new ProdList();
        Iterator iter = pList.getIterator();
        while(iter.has_next()) {
```

```
            ____________________________________;    //LINE 1
        }
    }
}
```

Identify the appropriate statement to fill in the blank at `LINE 1`, such that the output is:

```
P1001, Laptop
P1002, Smartphone
P1003, Smartwatch
```

- √ `((Printable)iter.get_next()).print()`
- ◯ `((Product)iter.get_next()).print()`
- ◯ `((ProdList)iter.get_next()).print()`
- ◯ `iter.get_next().print();`

15. Consider the Java code given below that prints the highest priced stock among a set of given `Stock` objects. From among the options, identify the appropriate function header for the function `printHighestPricedStock` that takes as input an array of `Stock` objects and prints the highest priced stock. [MSQ : 5 Marks]

```
import java.util.*;
interface Stock {
    public abstract double getPrice();
}
class AStock implements Stock {
    private double price;
    // Constructor
    // method getPrice() that returns price
}
class BStock implements Stock {
    private double price;
    // Constructor
    // method getPrice() that returns price
}
public class Test {
     // LINE 1: FUNCTION HEADER
    {
        // invokes method getPrice()
        // to print the value of highest priced stock
    }
    public static void main(String[] args) {
        Stock[] stocks = {
            new AStock(150.50),
            new BStock(200.75),
            new AStock(160.25)
        };
        printHighestPricedStock(stocks);
    }
}
```

Choose the correct option(s).

◯ `public static <T extends  AStock> void printHighestPricedStock(T[] items)`

√ `public static <T extends Stock> void printHighestPricedStock(T[] items)`

◯ `public static <T extends BStock> void printHighestPricedStock(T[] items)`

√ `public static void printHighestPricedStock(Stock[] items)`

16. Consider the code given below. [MCQ : 4 Marks]

```
import java.util.*;
class ZeroValueException extends Exception {
    public String toString() {
        return "Zero encountered during update";
    }
}
public class Test {
    public static void update(int[] array, int index) throws ZeroValueException
    {
        if (array[index] == 0) {
            throw new ZeroValueException();
        }
        array[index] = array[index] * 5;
    }
    public static void main(String[] args) {
        int[] arr = {1, -1, 0, 2, -2};
        try {
            for (int i = 0; i < arr.length; i++) {
                update(arr, i);
            }
        } catch (ZeroValueException e) {
            System.out.println(e);
        }
        for (int n : arr) {
            System.out.print(n + " ");
        }
    }
}
```

What will the output be?

◯ `Zero encountered during update`

◯ `5 -5 0 2 -2`

◯ `Zero encountered during update`
`5 -5`

√ `Zero encountered during update`
`5 -5 0 2 -2`

17. Method `Optional.ofNullable(T value)` returns an `Optional` that describes the specific value, if non-null; otherwise returns an `empty Optional`.
Based on this description, consider the code given below, and answer the question that follows. [MCQ : 4 Marks]

```
import java.util.*;
class Movie {
   HashMap<String, String> actors = new HashMap<>();
   public Movie() {
      actors.put("Action", "Akshay");
      actors.put("Comedy", "Kapil");
   }
   public String getActor(String genre) {
      return actors.get(genre);
   }
}
public class Test {
   public static void main(String[] args) {
      Optional<String> a1 = Optional.ofNullable(new Movie().getActor("Action"));
      Optional<String> a2 = Optional.ofNullable(new Movie().getActor("Thriller"));
      a1.ifPresent(n ->System.out.println(n.toUpperCase()));
      a2.ifPresent(n -> System.out.println(n.toUpperCase()));
   }
}
```

Choose the correct option.

- √ This program generates the output:
  `AKSHAY`
- ◯ This program terminates due to `NullPointerException` after printing the message:
  `AKSHAY`
- ◯ This program generates the output:
  `AKSHAY`
  `null`
- ◯ This program generates the output:
  `ACTION`
  `AKSHAY`

18. Consider the Java code given below. [MCQ : 4 Marks]

```
import java.io.*;
class HealthCard implements Serializable {
    private String cardNumber = "******";
    private transient String insuranceProvider = "Unknown";
    private String issueDate = "00/00";
    public HealthCard(String cN, String iP, String iD) {
        cardNumber = cN;
        insuranceProvider = iP;
        issueDate = iD;
    }
    public String toString() {
        return  cardNumber + ", " + insuranceProvider + ", " + issueDate;
    }
}
public class Test {
    public static void main(String[] args) throws Exception {
        var fos = new FileOutputStream("healthcard.txt");
        var os = new ObjectOutputStream(fos);
        os.writeObject(new HealthCard("H123456", "HInsurance", "03/24"));
        os.close();
        var fis = new FileInputStream("healthcard.txt");
        var ois = new ObjectInputStream(fis);
        HealthCard card = (HealthCard) ois.readObject();
        ois.close();
        System.out.println(card);
    }
}
```

What will the output be?

- ◯ `null, null, null`
- √ `H123456, null, 03/24`
- ◯ `H123456, Unknown, 03/24`
- ◯ `H123456, HInsurance, 03/24`
- ◯ `********, Unknown, 00/00`

19. Consider the Java code given below. [MCQ : 4 Marks]

```
import java.util.*;
import java.util.stream.*;
class Car {
    private String model;
    private double mileage;
    //Constructor to initialize instance variables
    public double getMileage() {
        return mileage;
    }
    public String toString() {
        return model;
    }
}
public class Test {
    public static void main(String[] args) {
        var carArr = new ArrayList<Car>();
        carArr.add(new Car("Toyota", 20.5));
        carArr.add(new Car("Ford", 25.3));
        carArr.add(new Car("Honda", 18.9));
        carArr.add(new Car("Chevrolet", 22.0));
        Map<Boolean, List<Car>> mileageMap;
        mileageMap = carArr.stream()
                  .collect(Collectors.partitioningBy(c -> c.getMileage() >= 22.0));
        System.out.println(mileageMap.get(false));
    }
}
```

Choose the correct option.

- ◯ This program generates the output: `[Ford, Chevrolet]`
- ◯ This program generates the output: `[Ford]`
- √ This program generates the output: `[Toyota, Honda]`
- ◯ This program generates the output: `[Toyota, Honda, Chevrolet]`

20. Consider the code given below. Assume that the file `food.txt` contains the following lines of text in it. [MCQ : 4 Marks]
A balanced diet is key to good health.
Food provides essential nutrients for the body.
Food preparation is an art form.

```
import java.io.*;
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        try {
            var in=new FileInputStream("food.txt");
            var scanner=new Scanner(in);   //LINE 1
            System.out.println("Data from file:");
            System.out.println(scanner.nextLine());
            System.out.println(scanner.next());
            System.out.println(scanner.nextLine());
        }
        catch (FileNotFoundException e) {
            System.out.println("File does not exist.");
        }
        catch (IOException e) {
            System.out.println("Error in writing a file.");
        }
    }
}
```

Choose the correct option.

◯ `LINE 1` generates `IOException`.

◯ This program generates the output:
```
Data from file:
A balanced diet is key to good health.
Food provides essential nutrients for the body.
Food preparation is an art form.
```

√ This program generates the output:
```
Data from file:
A balanced diet is key to good health.
Food
 provides essential nutrients for the body.
```

◯ This program generates the output:

```
Data from file:
A balanced diet is key to good health.
A
Food preparation is an art form.
```

21. Consider the Java code given below. [MSQ : 6 Marks]

```
class Pattern implements Runnable {
    boolean stopRequested = false;
    String[] pattern = {"One", "Two", "Three", "Four", "Five"};
    int index = 0;
    public void run() {
        while (!stopRequested) {
            System.out.print(pattern[index] + " ");
            index = (index + 1) % pattern.length;
        }
    }
    public void setStop(boolean stop) {
        stopRequested = stop;
    }
}
public class Test {
    public static void main(String[] args) throws InterruptedException {
        Pattern p = new Pattern();
        Thread t1 = new Thread(p);
        t1.start();
        p.setStop(true);
    }
}
```

Choose the correct option(s).

- ◯ The program will always generate the output: `One Two Three Four Five`
- ◯ The program will always generate the output: `One`
- √ The output can be `One` or `One Two` or `One Two Three` or `One Two Three Four` or `One Two Three Four Five` and can also cycle back and start again.
- √ The program may not generate any output.

22. Consider the Java code given below. [MSQ : 5 Marks]

```
class Stadium {
    int available = 1;
    public synchronized void bookSeat(int n, String name) {
        if (available >= n) {
            available = available - n;
            System.out.println(name + " booked " + n + " seat");
        } else {
            System.out.println(name + " cannot book " + n + " seat");
        }
    }
}
class SeatBooking implements Runnable {
    private Stadium s;
    private String name;
    private int n_seats;
    public SeatBooking(Stadium s, String n, int ns) {
        this.s = s;
        this.name = n;
        this.n_seats = ns;
    }
    public void run() {
        s.bookSeat(n_seats, name);
    }
}
public class ThreadTest {
    public static void main(String[] args) {
        Stadium obj = new Stadium();
        SeatBooking sb1 = new SeatBooking(obj, "Virat", 1);
        SeatBooking sb2 = new SeatBooking(obj, "Saniya", 1);
        Thread t1 = new Thread(sb1);
        Thread t2 = new Thread(sb2);
        t1.start();
        t2.start();
    }
}
```

Which of the following options is/are possible result/s of the above code?

√ `Saniya booked 1 seat`
`Virat cannot book 1 seat`

◯ `Saniya booked 1 seat`
`Virat booked 1 seat`

√ Virat booked 1 seat
Saniya cannot book 1 seat

◯ Virat cannot book 1 seat
Saniya cannot book 1 seat

23. Consider the Java code given below. [MSQ : 5 Marks]

```
import java.util.*;
import java.util.concurrent.*;
class Example extends Thread {
    Map cuMap;
    Example(Map m) {
        this.cuMap = m;
    }
    public void run() {
        cuMap.put("4","Four");
    }
}
public class Test {
    public static void main (String[] args) {
        Map<Integer, String> cuMap = new ConcurrentHashMap();
        Integer[] iarr = {1, 2, 3};
        String[] arr = {"One", "Two", "Three"};
        for(int i = 0; i < iarr.length; i++) {
            cuMap.put(iarr[i],arr[i]);
        }
        Example t = new Example(cuMap);
        t.start();
        Set s = cuMap.entrySet();
        Iterator itr = s.iterator();
        while(itr.hasNext()) {
            Map.Entry m = (Map.Entry)itr.next();
            System.out.println(m.getKey() + " => " + m.getValue());
        }
    }
}
```

Which of the following is **true** about the given code.

◯ This program may generate `ConcurrentModificationException`.

√ The program may generate the output:

```
1 => One
2 => Two
3 => Three
4 => Four
```

√ The program may generate the output:

```
1 => One
2 => Two
3 => Three
```

◯ The program always generate the output:

```
1 => One
2 => Two
3 => Three
4 => Four
```