Sample Paper

Test Summary

No. of Sections: 1
Total Questions: 15
Total Duration: 40 mins

Section 1 - CA3

Section Summary

No. of Questions: 15
Duration: 40 mins

Q1) Identify the correct replacement for the marked issue so that the program prints "Child" instead of "Parent".

Code Snippet

```
class Parent {
   void show() {
      System.out.println("Parent");
   }
}

class Child extends Parent {
   void show() {
      System.out.println("Child");
   }
}

public class Main {
   public static void main(String[] args) {
      Parent obj = new Child();
      obj.show(); // Issue
   }
}
```

Options

1) Use @Override annotation in Child class
2) Declare show() as final in Parent class
3) Make show() static in both Parent and Child classes
4) No change needed

Q2) Identify the correct replacement for the marked issue so that the program correctly ensures the resource is closed when an exception occurs.

Code Snippet

```
import java.io.*;

public class Main {
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new FileReader("data.txt"));
      try {
         System.out.println(br.readLine());
      } catch (IOException e) {
         System.out.println("Error reading file");
      } finally {
         br.close(); // Issue
      }
```

```
    }
}
```

Options

1) Use try-with-resources instead of explicit close()
2) Move br.close() inside the try block.
3) Surround br.close() with a try-catch
4) Use System.gc() before br.close()

Q3) Identify the correct replacement for the marked issue so that the program correctly prints "Sum: 45" instead of "Sum: 27".

Code Snippet

```
public class Main {
    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        System.out.println("Sum: " + sum(matrix, 0, 1, 2));
    }

    static int sum(int[][] arr, int... indices) {
        int total = 0;
        for (int i : indices) {
            for (int j = 0; j < arr[i].length - 1; j++) { // Issue
                total += arr[i][j];
            }
        }
        return total;
    }
}
```

Options

1) Replace arr[i].length - 1 with arr[i].length
2) Use for (int j : arr[i]) instead of for (int j = 0; j < arr[i].length - 1; j++)
3) Use Arrays.stream(arr[i]).sum(); instead of inner loop
4) Use sum(arr, indices) instead of sum(matrix, 1, 2, 3)

Q4) Identify the correct replacement for the marked issue so that the program always prints "Lambda Thread Running" before "Main Method Done".

Code Snippet

```
public class Main {
    public static void main(String[] args) { throws InterruptedException {
        Runnable r = () -> System.out.println("Lambda Thread Running");
        Thread t = new Thread(r);
        t.run(); // Issue
        System.out.println("Main Method Done");
    }
}
```

Options

1) Replace t.run(); with t.start();
2) Use t.start(); followed by t.join();
3) Declare t as static
4) Use synchronized on System.out.println()

Q5) Fill in the blanks to override toString() method. Output: Value: 100

Code Snippet

```
class Box {
    int value = 100;
    public String toString() {
```

```
        _____
    }
}
public class Main {
    public static void main(String[] args) {
        Box b = new Box();
        _____
    }
}
```

Q6) Fill in the blanks to print even numbers from 1 to 10. Output: 2 4 6 8 10

Code Snippet

```
public class Main {
    public static void main(String[] args) {
        _____
            if (i % 2 == 0)
                _____;
        _____
    }
}
```

Q7) Fill in the blanks to write to file. Effect: Creates file.txt with content ("File created")

Code Snippet

```
import java.io.FileWriter;
public class Main {
    public static void main(String[] args) {
        try (_____) {
            fw.write("File created");
        } catch (Exception e) {
            _____
        }
    }
}
```

Q8) Fill in the blanks to read input from console. Output: Enter name: Hello John

Code Snippet

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        _____
        System.out.print("Enter name: ");
        String name = sc.nextLine();
        _____
    }
}
```

Q9) Which of the following programs will print "Complete" without any compilation errors?

A.

```
class Alpha {
    void execute() { System.out.print("Processing "); }
}
class Beta extends Alpha {
    void execute() { super.execute(); System.out.println("Complete"); }
}
public class Main {
    public static void main(String[] args) {
        Alpha obj = new Beta();
        obj.execute();
```

```
    }
}
```

B.

```
class Alpha {
    void execute() { System.out.print("Processing "); }
}
class Beta extends Alpha {
    final void execute() { System.out.println("Complete"); }
}
public class Main {
    public static void main(String[] args) {
        Alpha obj = new Beta();
        obj.execute();
    }
}
```

C.

```
class Alpha {
    void execute() { System.out.print("Processing "); }
}
class Beta extends Alpha {
    void execute() { System.out.println("Complete"); }
}
public class Main {
    public static void main(String[] args) {
        Beta obj = new Beta();
        obj.execute();
    }
}
```

D.

```
class Alpha {
    void execute() { System.out.print("Processing "); }
}
class Beta extends Alpha {
    void execute(int x) { System.out.println("Complete"); }
}
public class Main {
    public static void main(String[] args) {
        Alpha obj = new Beta();
        obj.execute();
    }
}
```

Options

1) A, B, D
2) A, C
3) D, C
4) B, C, D

Q10) Which of the following programs will print "Task Complete" without any compilation errors?

A.
```
class Outer {
   static class Inner {
      void display() { System.out.print("Task "); }
   }
}
public class Main {
   public static void main(String[] args) {
      Outer.Inner obj = new Outer.Inner();
      obj.display();
      System.out.println("Completed");
   }
}
```

B.
```
class Outer {
   class Inner {
      void display() { System.out.print("Task "); }
   }
}
public class Main {
   public static void main(String[] args) {
      Outer.Inner obj = new Outer().new Inner();
      obj.display();
      System.out.println("Completed");
   }
}
```

C.
```
class Outer {
   private static class Inner {
      void display() { System.out.println("Task Completed"); }
   }
   public static void process() {
      Inner obj = new Inner();
      obj.display();
   }
}
public class Main {
   public static void main(String[] args) {
      Outer.process();
   }
}
```

D.
```
class Outer {
   class Inner {
      static void display() { System.out.println("Task Completed"); }
   }
}
```

```
public class Main {
   public static void main(String[] args) {
      Outer.Inner.display();
   }
}
```

Options

1) A, B, D
2) A, B, C
3) D, C
4) B, C, D

Q11) Which of the following programs will print Derived when executed successfully?

A.

```
class Base {
   void show() { System.out.print("Base "); }
}
class Derived extends Base {
   void show() { System.out.print("Derived"); }
}
public class Main {
   public static void main(String[] args) {
      Base b = new Derived();
      b.show();
   }
}
```

B.

```
class Base {
   void display() { System.out.print("Base "); }
}
class Derived extends Base {
   void display() { System.out.print("Derived"); }
}
public class Main {
   public static void main(String[] args) {
      Base b = new Derived();
      if (b instanceof Derived) {
        b.display();
      }
   }
}
```

C.

```
class Base {
   final void show() { System.out.print("Base "); }
}
class Derived extends Base {
   void show() { System.out.print("Derived"); }
}
public class Main {
```

```
    public static void main(String[] args) {
        Base b = new Derived();
        b.show();
    }
}
```

D.

```
class Base {
    void show() { System.out.print("Base "); }
}
class Derived extends Base {
    void show() { System.out.print("Derived"); }
}
public class Main {
    public static void main(String[] args) {
        Derived d = new Derived();
        d.show();
    }
}
```

Options

1) A, B, D
2) A, C
3) D, C
4) B, C, D

Q12) A text file named data.txt contains the following lines:
Java I/O is powerful
File handling is essential
Streams are useful

A developer wants to read the file and print only the lines that contain the word "essential". Which of the following code snippets will successfully compile and correctly achieve this?

A.

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("data.txt");
        BufferedReader br = new BufferedReader(fr);
        String content = br.lines().reduce("", (a, b) -> a + "\n" + b);
        System.out.println(content);
        br.close();
    }
}
```

B.

```
import java.nio.file.*;
import java.io.IOException;
public class Main {
    public static void main(String[] args) throws IOException {
        Files.lines(Paths.get("data.txt"))
```

```
            .filter(line -> line.contains("essential"))
            .forEach(System.out::println);
    }
}
```

C.

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("data.txt"));
        String line;
        while ((line = br.readLine()) != null) {
            if (line.contains("essential")) {
                System.out.println(line);
            }
        }
        br.close();
    }
}
```

D.

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws IOException {
        RandomAccessFile raf = new RandomAccessFile("data.txt", "rw");
        raf.seek(raf.length());
        raf.writeBytes("Adding new data to file\n");
        raf.close();
    }
}
```

Options

1) A, C, D
2) A, B
3) B, A, C
4) C, B

Q13) Arrange the code to write data to a file.

Code Snippet

```
import java.io.FileWriter;
class FileWrite {
    public static void main(String[] args) {
1.      catch (Exception e) {
2.          fw.write("Welcome to Java IO");
3.      }
4.      try (FileWriter fw = new FileWriter("output.txt")) {
5.          e.printStackTrace();
6.      }
    }
}
```

Options

1) 1 2 3 4 5 6

2) 4 2 3 1 5 6
3) 2 3 4 1 5 6
4) 2 4 1 3 5 6

Q14) Arrange the code to read input from console.

Code Snippet

```
import java.util.Scanner;
class ConsoleInput {
   public static void main(String[] args) {
1.    System.out.println("Hello " + name);
2.    System.out.print("Enter name: ");
3.    String name = sc.nextLine();
4.    Scanner sc = new Scanner(System.in);
   }
}
```

Options

1) 2 1 3 4
2) 2 4 3 1
3) 2 3 4 1
4) 2 4 1 3

Q15) Arrange the code to print whether a number is positive or negative using if-else.

Code Snippet

```
class Test {
   public static void main(String[] args) {
1.    int num = -8;
2.    if (num >= 0) {
3.       System.out.println("Negative");
4.    }
5.    }
5.       System.out.println("Positive");
6.    else {
   }
}
```

Options

1) 1 2 5 6 3 4
2) 1 2 6 5 3 4
3) 1 5 6 4 3 2
4) 1 2 3 4 5 6