MODULE 1

---

1. Difference between 1D and Multidimensional Array:

- A 1D array stores data in a single row (linear form).

  Example: int arr[] = {1, 2, 3};

- A multidimensional array stores data in a table form (rows and columns).

  Example: int arr[][] = {{1, 2}, {3, 4}};

---

2. Irregular Array in Java:

An irregular (jagged) array is a multidimensional array where each row can have a different number of columns.

Example:

int arr[][] = { {1, 2}, {3, 4, 5} };

---

3. Purpose of length() method in String class:

The length() method returns the number of characters in a string.

Example:

String s = "Java";
System.out.println(s.length()); // Output: 4

---

4. Two differences between Vector and ArrayList:

1. Synchronization: Vector is synchronized; ArrayList is not.
2. Growth: Vector increases size by 100%, ArrayList by 50%.

---

5. String Literal in Java:

A string literal is a sequence of characters enclosed in double quotes, e.g., "Hello".

String literals are stored in the String constant pool in memory to save space and reuse existing strings.

---

6. Output of the given program:

System.out.println("Hello" + 5 + 10);

Output: Hello510

(→ Because "Hello" + 5 forms a string, then concatenates with 10.)

---

7. Java statement using String.join():

String s = String.join(" ", "JAVA", "PROGRAMMING");

---

8. Reverse contents of StringBuffer object sb:

sb.reverse();

---

9. Use of super keyword:

- Used to call the parent class constructor or access parent class members hidden by subclass.

  Example: super.display();

---

10. Uses of final keyword:

1. final variable: value cannot be changed.
2. final method: cannot be overridden.
3. final class: cannot be inherited.

MODULE 2

---

1. What is an interface in Java:

An interface in Java is a collection of abstract methods and constants that defines a contract which implementing classes must follow.

Example:

interface Shape { void draw(); }

---

2. Difference between multiple and single inheritance:

- Single inheritance: A class inherits from one superclass.
- Multiple inheritance: A class inherits from more than one parent (not allowed with classes, but achieved through interfaces in Java).

---

3. Keyword used to implement an interface:

The keyword implements is used.

Example:

```
class Circle implements Shape { ... }
```

---

4. Can an interface extend another interface? Give an example:

Yes, an interface can extend another interface.

Example:

```
interface A { void show(); }
interface B extends A { void display(); }
```

---

5. What is a nested interface:

A nested interface is an interface declared inside a class or another interface.

Example:

```
class Outer {
  interface Inner { void msg(); }
}
```

---

6. Purpose of the package keyword:

The package keyword is used to group related classes and interfaces together, helping in organizing code and avoiding name conflicts.

---

7. How to import a user-defined package:

Use the import statement before the class declaration.

Example:

import mypackage.MyClass;

---

8. Difference between import and static import:

- import: Used to access classes from another package.
- static import: Used to access static members directly without class name.

  Example: import static java.lang.Math.*;

---

9. Two advantages of using packages:

1. Helps in organizing classes logically.
2. Avoids naming conflicts between classes.

---

10. Significance of the File class in Java I/O:

The File class represents file and directory paths in Java. It is used to create, delete, and get information about files and directories.

MODULE 3

---

1. What happens when an exception is not caught in Java:

If an exception is not caught, the JVM terminates the program abnormally and displays an error message and stack trace.

Example:

int a = 10 / 0; // Uncaught ArithmeticException → program stops

---

2. Difference between checked and unchecked exceptions:

- Checked exceptions: Checked at compile-time (e.g., IOException, SQLException).

- Unchecked exceptions: Checked at runtime (e.g., NullPointerException, ArithmeticException).

---

3. Purpose of the finally block (with analogy):

The finally block is used to execute important cleanup code (like closing files) whether or not an exception occurs.

Analogy: Like locking the door (cleanup) whether or not you finish your work inside a room.

---

4. throw and throws keywords:

- throw – used to manually throw an exception.

  Example: throw new ArithmeticException("Divide by zero");

- throws – used in method signature to declare possible exceptions.

  Example: void read() throws IOException { }

---

5. Exception for 0 withdrawal amount in banking system:

A user-defined exception (e.g., InvalidAmountException) can be created to handle this case.

Example:

if(amount == 0) throw new InvalidAmountException("Amount cannot be zero");

---

6. Significance of start() method in multithreading:

The start() method starts a new thread and calls the run() method internally. It allows concurrent execution of threads.

---

7. Define an inner class and its usefulness:

An inner class is a class defined inside another class.

It helps in encapsulation and modular design by grouping related logic together.

Example:

```
class Outer {
  class Inner { void display() { } }
}
```

---

8. Lambda expression and its role in event handling:

A lambda expression provides a shorter syntax for implementing functional interfaces (like event listeners).

Example:

```
button.addActionListener(e -> System.out.println("Clicked!"));
```

It simplifies event handling by avoiding anonymous inner classes.
