2023 Exam

Que 1 :

1). Why is java called dynamic programming language ?

Solution: Java is a dynamic programming language because it can modify and adapt itself during runtime. This feature allows developers to make changes to a program without having to recompile it completely.

Here are some reasons why Java is considered a dynamic programming language:

- Dynamic features
  Java supports dynamic features like dynamic proxies, which have been available since JDK version 1.3.

- Dynamic type checking
  Java allows some dynamic type checking of values, especially for receivers of virtual or interface method calls.

- Loads class files at runtime
  Java loads class files at runtime, which means that the same code can be run on multiple platforms as long as the JDK is installed.

- Adapts to its environment
  Java can adapt to its environment by allowing programmers to link new class libraries, objects, and methods dynamically.

- Interoperability
  Java's dynamic capabilities allow it to interact with other languages and environments.

2). What is the advantage of using packages in java ?

Code Organization: Packages group related classes and interfaces, making the codebase more structured and easier to manage.

Encapsulation: Packages help control access to classes and members, supporting data hiding and security.

Namespace Management: Packages prevent naming conflicts by isolating classes into unique namespaces.

Modularity and Reusability: Packages allow for better modular design, making code easier to reuse and maintain.

Versioning and Dependency Management: Packages help manage dependencies and versions efficiently, especially with tools like Maven or Gradle.

Distribution and Deployment: Packages simplify the creation of JAR files, improving the distribution and deployment of applications.

Team Collaboration: Packages enable team members to work on separate modules, reducing conflicts and improving collaboration.

Code Navigation and IDE Support: Packages improve code discoverability and navigation, enhancing the development experience in IDEs.

3). What do mean of catch? How it is used in exception in java?

In Java, **catch** is a keyword used in exception handling to define a block of code that handles exceptions (errors) that occur during the execution of a program. The catch block is part of a **try-catch** structure, where:

- The **try** block contains code that might throw an exception.
- The **catch** block is used to handle the exception if one occurs.

Purpose of catch:

The catch block catches exceptions thrown by the try block and allows the program to continue executing after handling the exception, rather than terminating abruptly.

Syntax of catch in Java:

```
try {
   // Code that might throw an exception
} catch (ExceptionType e) {
   // Code to handle the exception
}
```

- **ExceptionType**: This specifies the type of exception that the catch block will handle (e.g., IOException, ArithmeticException, etc.).
- **e**: This is the exception object, which provides details about the exception that was thrown.
-

Example of catch in action:

```
public class Example {
    public static void main(String[] args) {
        try {
            // Code that might throw an exception
            int result = 10 / 0;  // This will throw an ArithmeticException
        } catch (ArithmeticException e) {
            // This block handles the exception
            System.out.println("Error: Division by zero is not allowed.");
        }

        System.out.println("Program continues after handling the
exception.");
    }
}
```

4). What is Garbage Collection in Java?

Garbage Collection (GC) in Java automatically reclaims memory by removing unreachable objects, helping manage memory efficiently. It works in phases like **marking**, **sweeping**, and **compacting** to free up space.

Key Points:

1. **Automatic Memory Management**: Java GC eliminates the need for manual memory deallocation by removing unused objects.
2. **Heap Structure**: Objects are stored in the heap, with the Young and Old generations, and are cleaned through Minor and Major GCs.
3. **Different GC Types**: Java offers various GC algorithms (Serial, Parallel, G1, CMS, ZGC) for different performance needs.

5). Give the output of following program :

class Output {
public static void main(String args])
{int a = 2, b = 5, c = 4;
a = 6; b >>= 1; c <<= 1;
System.out.println(a+""+ b+" " + c); 3}

Let's break down the program and explain how the operations work step by step:

Code:

java
Copy code

```java
class Output {
    public static void main(String args[]) {
        int a = 2, b = 5, c = 4;
        a = 6;
        b >>= 1;
        c <<= 1;
        System.out.println(a + "" + b + " " + c);
    }
}
```

Output - 62   8

6). Local vs Global Applets

In Java, **applets** are small programs that run within a web browser or an applet viewer. They can be classified into two types based on where they are loaded and executed: **local applets** and **remote applets**. Here's a breakdown of the differences:

1. Local Applet

- **Definition**: A **local applet** is one that is stored and executed from the local file system of the user's machine.
- **Location**: It is usually stored as a .class file on the local machine and run from there (i.e., executed from the user's local environment).
- **Loading**: It doesn't require downloading over the network, as it's already present on the user's system.
- **Security**: Local applets can be subject to more relaxed security restrictions, since they are loaded from a trusted local file system.

- **Use Cases**: Local applets are used for offline applications or testing purposes where the applet doesn't need to be fetched from a server.
- **Example**: An applet stored in the local directory of a system and run through a local browser or applet viewer.

java
Copy code
```
<applet code="LocalApplet.class" width="300" height="200">
</applet>
```
2. Remote Applet

- **Definition**: A **remote applet** is one that is stored on a web server and is downloaded and executed in a user's web browser when requested.
- **Location**: It is stored on a remote server and fetched over the internet when the user visits a web page that includes the applet.
- **Loading**: The applet is loaded dynamically over a network (from the web server) when the user accesses the HTML page containing the applet tag.
- **Security**: Remote applets are subject to stricter **security restrictions** due to the potential risks of downloading and executing code from untrusted sources. This is handled through the **Java security manager** and **sandboxing**.
- **Use Cases**: Remote applets are typically used for interactive web-based applications, such as online games, tools, and utilities embedded in web pages.
- **Example**: An applet fetched from a web server and executed in the user's browser.

html
Copy code
```
<applet code="RemoteApplet.class" archive="http://example.com/
applets/RemoteApplet.jar" width="300" height="200">
</applet>
```
Key Differences:

Feature	Local Applet	Remote Applet
Location	Stored on the local machine	Stored on a remote web server

Loading	Loaded directly from the local file system	Loaded over the internet via the applet tag in HTML
Security	Fewer security restrictions (local environment)	Subject to stricter security (sandbox)
Network Dependency	Does not require internet access	Requires internet to fetch the applet
Use Case	Offline use or local testing	Web-based interactive applications

Summary:

- **Local applets** are loaded and executed from the local file system, with fewer security restrictions, while **remote applets** are fetched from a web server and have stricter security policies due to the risks of executing code from an external source.

7). Explain tokens in java ?

In Java, **tokens** are the smallest individual units of a program. They are the building blocks of the Java syntax and are categorized as keywords, operators, literals, and identifiers. Here are four main types of tokens in Java:

1. Keywords

- **Definition**: Keywords are reserved words in Java that have a predefined meaning and cannot be used as identifiers (variable names, method names, etc.).
- **Examples**: int, class, if, else, public, private, static, void
- **Usage**: Keywords define the structure and behavior of the language.java

  public class Example {
-    public static void main(String[] args) {
-       int a = 5; // 'int' and 'public' are keywords
-    }
- }

2. Identifiers

- **Definition**: Identifiers are names given to various program elements such as classes, methods, variables, and packages.
- **Rules**: Identifiers must begin with a letter (A-Z or a-z), underscore (_), or dollar sign ($), followed by letters, digits (0-9), underscores, or dollar signs. They cannot be Java keywords.
- **Examples**: sum, totalAmount, MyClassjava
  int totalAmount = 100; // 'totalAmount' is an identifier

3. Literals

- **Definition**: Literals are fixed values used directly in the code. They represent constant values of a specific type (integer, floating-point, boolean, etc.).
- **Examples**: 10, 3.14, 'A', "Hello", true, false
- **Usage**: Literals are used to represent data in expressions and assignments.java
  int num = 100;   // '100' is an integer literal
- double pi = 3.14; // '3.14' is a floating-point literal

4. Operators

- **Definition**: Operators are symbols used to perform operations on variables or values. They manipulate data and variables in expressions.
- **Types**: Arithmetic operators, relational operators, logical operators, assignment operators, etc.
- **Examples**: +, -, *, /, =, ==, &&, ||java
  int x = 10;
- int y = 5;
- int sum = x + y; // '+' is an arithmetic operator

These tokens are essential components in Java programming, allowing the language to process and execute instructions effectively.

Que 2.