**Guru Tegh Bahadur Institute of Technology**

**IT\_1- 4th semester**

**Subject Name: Programming in Java**

**Subject Code: CIC212**

**Unit-I**

**MCQ Based question:**

1. **Which component of the Java Virtual Machine (JVM) manages memory allocation for objects created during program execution?**

A) Stack  
B) Heap  
C) Registers  
D) Cache

**Answer: B) Heap**

1. **What is the primary purpose of garbage collection in Java?**
   1. To prevent memory leaks  
      B) To optimize CPU usage  
      C) To minimize disk space usage  
      D) To speed up program execution

**Answer: A) To prevent memory leaks**

1. **In the context of JVM architecture, what is the purpose of the stack?**

A) To store method invocations and local variables  
B) To store dynamically allocated objects  
C) To manage memory allocation for static variables  
D) To handle input/output operations

**Answer: A) To store method invocations and local variables**

1. **Which model does Java use to provide a secure environment for executing untrusted code?**

A) Firewall model  
B) Containerization model  
C) Sandboxing model  
D) Virtualization model

**Answer: C) Sandboxing model**

1. **What happens when the JVM encounters an OutOfMemoryError?**

A) The JVM increases the heap size automatically.  
B) The JVM shuts down gracefully.  
C) The JVM terminates the application.  
D) The JVM triggers garbage collection.

**Answer: C) The JVM terminates the application.**

**Long Type Question:**

**Q.1** **Explain the characteristics of Java as a programming language and how they contribute to its platform independence and portability.**

Answer: Java is a platform-independent and portable language due to several key characteristics. First, Java is compiled into bytecode, which is platform-neutral and can be executed on any device with a Java Virtual Machine (JVM). Second, Java follows the principle of "write once, run anywhere," meaning that code written in Java can run unchanged on any platform with a compatible JVM. Third, Java has automatic memory management through garbage collection, which ensures efficient memory usage across different platforms.

**Q.2 Describe the role of the heap in Java memory management and explain how garbage collection works to reclaim memory on the heap.**

Answer: In Java, the heap is a region of memory used for dynamic memory allocation to objects at runtime. Objects created by the new keyword are allocated memory on the heap. Garbage collection is the process by which the JVM automatically identifies and removes objects from the heap that are no longer reachable or in use by the program. This is achieved by periodically scanning the heap for unreferenced objects and reclaiming their memory. Garbage collection helps prevent memory leaks and ensures efficient memory usage in Java programs.

**Q.3** **Discuss the architecture of the Java Virtual Machine (JVM), including its components and their roles in executing Java programs.**

Answer: The JVM is a virtual machine that provides an execution environment for Java bytecode. It consists of several components, including the class loader subsystem, runtime data areas (such as the method area, heap, and stack), execution engine, and native method interface. The class loader subsystem loads class files into memory, the runtime data areas store program data and instructions, the execution engine interprets bytecode or compiles it into native machine code for execution, and the native method interface allows Java code to interact with native libraries. Together, these components enable the JVM to execute Java programs in a platform-independent manner.

**Q.4 Explain the role of class loaders in Java, including their types and how they contribute to the dynamic loading and linking of classes at runtime.**

Answer: Class loaders are responsible for loading Java class files into memory at runtime. There are three types of class loaders in Java: the bootstrap class loader, extension class loader, and application class loader. The bootstrap class loader loads core Java classes from the bootstrap classpath, the extension class loader loads classes from the extension classpath, and the application class loader loads classes from the application classpath. Class loaders enable dynamic loading and linking of classes at runtime, allowing Java programs to load and use classes dynamically as needed.

**Q.5** **Discuss the security aspects of Java, including its sandbox model and how it provides a secure execution environment for untrusted code.**

Answer: Java's security model is based on the principle of sandboxing, which restricts the actions that untrusted code can perform to prevent malicious behavior. The Java sandbox model achieves this by imposing restrictions on untrusted code, such as preventing access to the file system, network, and native libraries. Additionally, Java's security manager enforces a security policy that specifies the permissions granted to code based on its origin and other factors. By running untrusted code within a controlled environment and enforcing security policies, Java provides a secure execution environment that protects against unauthorized access and malicious attacks.

**Short type Question:**

1. **What are two characteristics of Java that contribute to its platform independence?**

Answer: Java bytecode and the "write once, run anywhere" principle.

1. **What is the role of the heap in Java memory management?**

Answer: The heap is used for dynamic memory allocation to objects at runtime.

1. **Briefly explain how garbage collection works in Java.**

Answer: Garbage collection automatically identifies and removes objects from the heap that are no longer in use by the program.

1. **Name one component of the Java Virtual Machine (JVM) architecture and its role.**

Answer: Class loader subsystem - loads class files into memory at runtime.

1. **What is the primary purpose of the sandbox model in Java's security architecture?**

Answer: The sandbox model restricts the actions of untrusted code to prevent malicious behaviour and unauthorized access.

**Unit-II**

**MCQ Based Question:**

**1.** **Which of the following is NOT a primitive data type in Java?**

A) int  
B) float  
C) String  
D) boolean

**Answer: C) String**

**2**. **Which keyword is used to declare a constant value in Java?**

A) final  
B) static  
C) const  
D) volatile

**Answer: A) final**

**3.** **Which of the following is a wrapper class in Java for the primitive type int?**

A) Integer  
B) Float  
C) String  
D) Boolean

**Answer: A) Integer**

**4.** **Which class in Java provides a resizable array of characters and acts as a mutable sequence of characters?**

A) String  
B) StringBuilder  
C) StringBuffer  
D) Array

**Answer: C) StringBuffer**

**5.** **Which method is called first in the life cycle of an applet?**

A) destroy()  
B) init()  
C) start()  
D) stop()

**Answer: B) init()**

**Long Type Question:**

1. **Explain the concept of wrapper classes in Java and provide an example of when they might be used in a program.**

Answer: Wrapper classes in Java are used to convert primitive data types into objects. They provide methods to manipulate the data stored in these objects. For example, the Integer wrapper class is used to convert an int primitive data type into an object. This can be useful when working with collections, as collections in Java can only store objects, not primitives. So, if you need to store integers in a collection, you would use the Integer wrapper class.

1. **Discuss the significance of the throws clause in Java methods and how it relates to exception handling.**

Answer: The throws clause in Java methods is used to declare that a method may throw certain types of exceptions. This alerts the caller of the method that they need to handle or propagate those exceptions. It is part of Java's exception handling mechanism, which allows for the graceful handling of errors and exceptional conditions that may occur during program execution. By specifying the types of exceptions that a method may throw with the throws clause, Java provides a way for developers to handle errors at appropriate levels in their code.

1. **Explain the concept of anonymous inner classes in Java and provide a scenario where they might be used.**

Answer: Anonymous inner classes in Java are inner classes without a name. They are defined and instantiated at the same time. Anonymous inner classes are useful when you need to override methods of a class or implement an interface without explicitly creating a separate class. For example, you might use an anonymous inner class when implementing event listeners in graphical user interface (GUI) programming. Instead of creating a separate class for each listener interface implementation, you can define and instantiate the class inline, making the code more concise and readable.

1. **Discuss the life cycle of an applet in Java, including the methods that are called during each phase and their respective purposes.**

Answer: The life cycle of an applet in Java consists of four methods:

* + init(): This method is called when the applet is initialized, typically once per applet instance. It is used to perform any initialization tasks, such as setting up GUI components or initializing variables.
  + start(): This method is called after the init() method and whenever the applet is restarted. It is used to start any background tasks or animations.
  + stop(): This method is called when the applet is stopped, such as when the user navigates away from the page containing the applet. It is used to stop any background tasks or animations.
  + destroy(): This method is called when the applet is destroyed, typically when the web page containing the applet is closed. It is used to release any resources or clean up any state used by the applet.

1. **Explain the concept of inheritance in Java and provide an example of how it can be implemented in a program.**

Answer: Inheritance in Java allows a class to inherit properties and behaviours from another class, known as the superclass. The subclass inherits all the non-private members (fields and methods) of the superclass. This promotes code reuse and allows for the creation of hierarchical relationships between classes. For example, you might have a superclass Vehicle with properties and methods common to all vehicles, and subclasses Car and Motorcycle that inherit from Vehicle and add their own specific properties and methods. This allows you to write generic code that can work with any type of vehicle, while still allowing for specialization in subclasses.

**Short Type Question:**

Q.1 **What is the purpose of a wrapper class in Java?**

Answer: Wrapper classes are used to convert primitive data types into objects and provide utility methods for manipulation.

Q.2 **Explain the role of the final keyword in Java variables.**

Answer: The final keyword in Java variables indicates that the variable's value cannot be changed once initialized.

Q.3 **What is the primary purpose of using anonymous inner classes in Java?**

Answer: Anonymous inner classes are used to implement interfaces or extend classes inline without explicitly defining a named class.

Q.4 **Briefly describe the throws clause in Java methods.**

Answer: The throws clause in Java methods is used to declare that a method may throw certain types of exceptions and delegates the responsibility of handling those exceptions to the caller.

Q.5 **What is the first method called in the life cycle of an applet?**

Answer: The init() method is called first in the life cycle of an applet, which initializes the applet and its resources

Top of Form

Bottom of Form

**Unit-III**

**MCQ Based Question**

Q.1 **Which layout manager arranges components in a grid-like structure with rows and columns?**

A) Border layout  
B) Grid layout  
C) Flow layout  
D) Card layout

**Answer: B) Grid layout**

Q.2 **Which class in Java AWT is used as a base class for all the user interface elements?**

A) Component  
B) Container  
C) Frame  
D) Panel

**Answer: A) Component**

Q.3 **What does AWT stand for in Java programming?**

A) Abstract Window Toolkit  
B) Advanced Web Technologies  
C) Asynchronous Web Toolkit  
D) Application Window Toolkit

**Answer: A) Abstract Window Toolkit**

Q.4 **Which event listener interface is used to handle mouse events in Java AWT?**

A) ActionListener  
B) MouseListener  
C) KeyListener  
D) WindowListener

**Answer: B) MouseListener**

Q.5 **Which layout manager divides the container into five regions: North, South, East, West, and Center?**

A) Flow layout  
B) Grid layout  
C) Border layout  
D) Card layout

**Answer: C) Border layout**

**Long Type Question:**

Q.1 **Explain the concept of threads in Java and discuss their significance in multitasking and concurrent programming. Provide an example of when using threads might be beneficial in a Java program.**

Answer: Threads in Java are lightweight processes that execute independently and concurrently within a single program. They allow for multitasking, where multiple tasks can execute simultaneously, improving program responsiveness and performance. Threads are beneficial in scenarios where tasks can be executed concurrently without dependencies on each other. For example, in a web server application, using threads to handle multiple client requests simultaneously can improve throughput and responsiveness.

Q.2 **Describe the role of AWT components and the container class in Java GUI programming. How do they contribute to creating graphical user interfaces?**

Answer: AWT (Abstract Window Toolkit) components are the building blocks of graphical user interfaces in Java. They represent visual elements such as buttons, labels, text fields, etc., that users interact with. The container class in Java serves as a parent for AWT components and provides a space to organize and layout these components on the screen. By adding AWT components to containers and arranging them using layout managers such as Border Layout or Grid Layout, developers can create visually appealing and functional GUIs.

Q.3 **Explain the differences between Border Layout and Grid Layout in Java AWT. When would you choose one over the other in GUI design?**

Answer: Border Layout and Grid Layout are both layout managers in Java AWT but have different approaches to arranging components. Border Layout divides the container into five regions (North, South, East, West, Center) and places components in these regions, while Grid Layout arranges components in a grid-like structure with rows and columns. Developers would choose Border Layout when they want components to occupy specific regions of the container, such as placing a navigation bar in the North region and content in the Center region. Grid Layout would be chosen when a more structured layout with rows and columns is desired, such as arranging a set of buttons in a calculator application.

Q.4 **Discuss the concept of event handling in Java AWT and the role of event listeners. What are class listeners and adapters, and how do they simplify event handling in Java GUI programming?**

Answer: Event handling in Java AWT involves responding to user actions such as mouse clicks or key presses. Event listeners are interfaces that define methods to handle specific types of events. Class listeners are classes that implement these listener interfaces to handle events. Adapters are classes that provide default implementations of listener interfaces, allowing developers to selectively override only the methods they need. Class listeners and adapters simplify event handling by reducing the amount of boilerplate code required to implement event listeners, making GUI programming more concise and manageable.

**Q.5 Explain the concept of mouse events and window events in Java AWT. Provide examples of when each type of event might occur and how they can be handled in a Java program.**

Answer: Mouse events in Java AWT occur when the user interacts with the mouse, such as clicking or dragging. Examples include mouseClicked, mousePressed, and mouseReleased events. Window events occur when the state of a window changes, such as opening, closing, or resizing. Examples include windowOpened, windowClosing, and windowResized events. These events can be handled in a Java program by implementing corresponding event listener interfaces (e.g., MouseListener for mouse events, WindowListener for window events) and overriding their methods to define custom behavior in response to the events.

**Short Type Question:**

1. **What are threads in Java, and what is their primary purpose?**

Answer: Threads in Java are lightweight processes that enable multitasking and concurrent execution within a single program. Their primary purpose is to improve program responsiveness and performance by allowing multiple tasks to execute simultaneously.

1. **Name two types of layout managers in Java AWT for arranging GUI components.**

Answer: Two types of layout managers in Java AWT are Border Layout and Grid Layout.

1. **What is the role of event listeners in Java GUI programming?**

Answer: Event listeners in Java GUI programming are used to handle user interactions such as mouse clicks, key presses, and window events. They listen for specific events and execute corresponding code when those events occur.

1. **What is the purpose of using adapters in Java event handling?**

Answer: Adapters in Java event handling provide default implementations for event listener interfaces, allowing developers to implement only the methods they need. This simplifies event handling by reducing the amount of boilerplate code required.

1. **Give an example of a mouse event and a window event in Java AWT.**

Answer: An example of a mouse event is mouseClicked, which occurs when a mouse button is clicked. An example of a window event is windowClosing, which occurs when a window is about to Top of Form

**Unit-IV**

**Bottom of Form**

**MCQ Based question:**

Q.1 **Which Java class is used for reading primitive data types from an input stream?**

A) InputStream  
B) DataInputStream  
C) InputStreamReader  
D) BufferedReader

**Answer: B) DataInputStream**

Q.2 **Which Java interface provides a way to navigate through a collection of objects one by one?**

A) Set  
B) Iterator  
C) List  
D) Enumeration

**Answer: B) Iterator**

Q.3 **Which Java class provides a way to read from and write to a file in a random access manner?**

A) FileReader  
B) BufferedWriter  
C) RandomAccessFile  
D) FileInputStream

**Answer: C) RandomAccessFile**

Q.4 **Which Java API is used for connecting to databases such as MS Access, Oracle, or MS-SQL?**

A) Swing  
B) AWT  
C) JDBC  
D) Servlet

**Answer: C) JDBC**

Q.5 **Which Java collection class does not allow duplicate elements?**

A) List  
B) Set  
C) Map  
D) Vector

**Answer: B) Set**

**Long type Question:**

1. **Explain the concept of Input/Output streams in Java and how they are used for handling data input and output operations. Provide examples of different types of streams and their typical use cases.**

Answer: In Java, Input/Output (I/O) streams are used for reading data from or writing data to various sources, such as files, network connections, or memory buffers. Streams provide a convenient way to handle sequential data transfer in Java programs. Examples of input streams include FileInputStream for reading from files and ObjectInputStream for reading serialized objects. Examples of output streams include FileOutputStream for writing to files and ObjectOutputStream for writing serialized objects. These streams can be chained together using stream filters to perform additional processing on the data, such as buffering or compression.

1. **Discuss the role of JDBC in Java programming and how it enables database connectivity with various database management systems (DBMS) such as MS Access, Oracle, and MS-SQL. Provide an overview of the JDBC architecture and its key components.**

Answer: JDBC (Java Database Connectivity) is a Java API for connecting Java applications to relational databases. It provides a standard interface for Java applications to interact with different database management systems (DBMS) such as MS Access, Oracle, and MS-SQL. The JDBC architecture consists of three key components: the JDBC API, JDBC driver manager, and JDBC drivers. The JDBC API provides classes and interfaces for executing SQL queries, retrieving results, and managing database connections. The JDBC driver manager is responsible for loading and managing JDBC drivers, which are platform-specific implementations that handle communication with a particular DBMS. By using JDBC, Java developers can write database-independent code that can connect to and interact with various databases.

1. **Explain the concept of JNI (Java Native Interface) in Java programming and discuss its role in enabling interaction between Java code and native code written in languages such as C or C++. Provide an example scenario where JNI might be used in a Java program.**

Answer: JNI (Java Native Interface) is a Java programming framework that enables Java code to call and be called by native applications and libraries written in languages such as C or C++. JNI provides a way for Java applications to access platform-specific features or leverage existing native code libraries. An example scenario where JNI might be used is in performance-critical applications where certain operations need to be implemented in native code for improved efficiency. For instance, image processing algorithms written in C or C++ might be accessed from a Java application using JNI to achieve better performance.

1. **Discuss the concept of collections in Java and provide an overview of different collection interfaces and classes available in the Java Collections Framework (JCF). Explain the differences between sets, lists, and maps, and provide examples of when each type of collection might be used.**

Answer: Collections in Java are objects that group multiple elements into a single unit. The Java Collections Framework (JCF) provides a set of interfaces and classes for working with collections in Java programs. Some key interfaces in the JCF include Set, List, Map, and Collection. Sets represent unordered collections of unique elements, lists represent ordered collections of elements, and maps represent collections of key-value pairs. Examples of when each type of collection might be used include using sets to store a unique set of items, lists to maintain elements in a specific order, and maps to associate keys with values for efficient lookup.

1. **Explain the concept of iterators in Java and how they are used to traverse elements in collections. Provide examples of different types of iterators available in Java and describe their characteristics and typical use cases.**

Answer: Iterators in Java are objects that provide a way to traverse the elements in a collection sequentially. They allow for iterating over the elements of a collection without exposing the internal structure of the collection. Some common types of iterators in Java include the Iterator interface, which provides a generic way to traverse elements in a collection, and specialized iterators such as ListIterator, which allows bidirectional traversal of elements in a list. Iterators are typically used in conjunction with collection classes to perform operations such as iterating over the elements of a list or set and performing actions on each element.

Top of Form

Bottom of Form

**Short Type Question:**

Q.1**What is the purpose of a PrintStream in Java I/O?**

Answer: PrintStream is used for printing formatted representations of objects to an output stream.

Q.2**What is the primary function of JDBC in Java programming?**

Answer: JDBC (Java Database Connectivity) is used for connecting Java applications to relational databases for data manipulation and retrieval.

Q.3**Name one method of the RandomAccessFile class in Java.**

Answer: One method of the RandomAccessFile class is seek(long pos) which sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

Q.4**What does JNI stand for, and what is its purpose in Java programming?**

Answer: JNI stands for Java Native Interface. It enables Java code to call and be called by native applications and libraries written in languages such as C or C++.

Q.5**Which Java collection class guarantees no duplicate elements and no specific order?**

Answer: Set interface and its implementations (e.g., HashSet, TreeSet) guarantee no duplicate elements and no specific order

Top of Form

Bottom of Form

Top of Form

Bottom of Form