IGNOU | BCA

2023-2024

FREE

SOLVED ASSIGNMENT

All In One

MCS-024

Object-Oriented Technologies and Java Programming

Prepared by

ISM

730-339-3555

www.ignoustudymentor.com

@ism_ignoustudymentor_

ISM - IGNOU STUDY MENTOR

Youtube Channel and

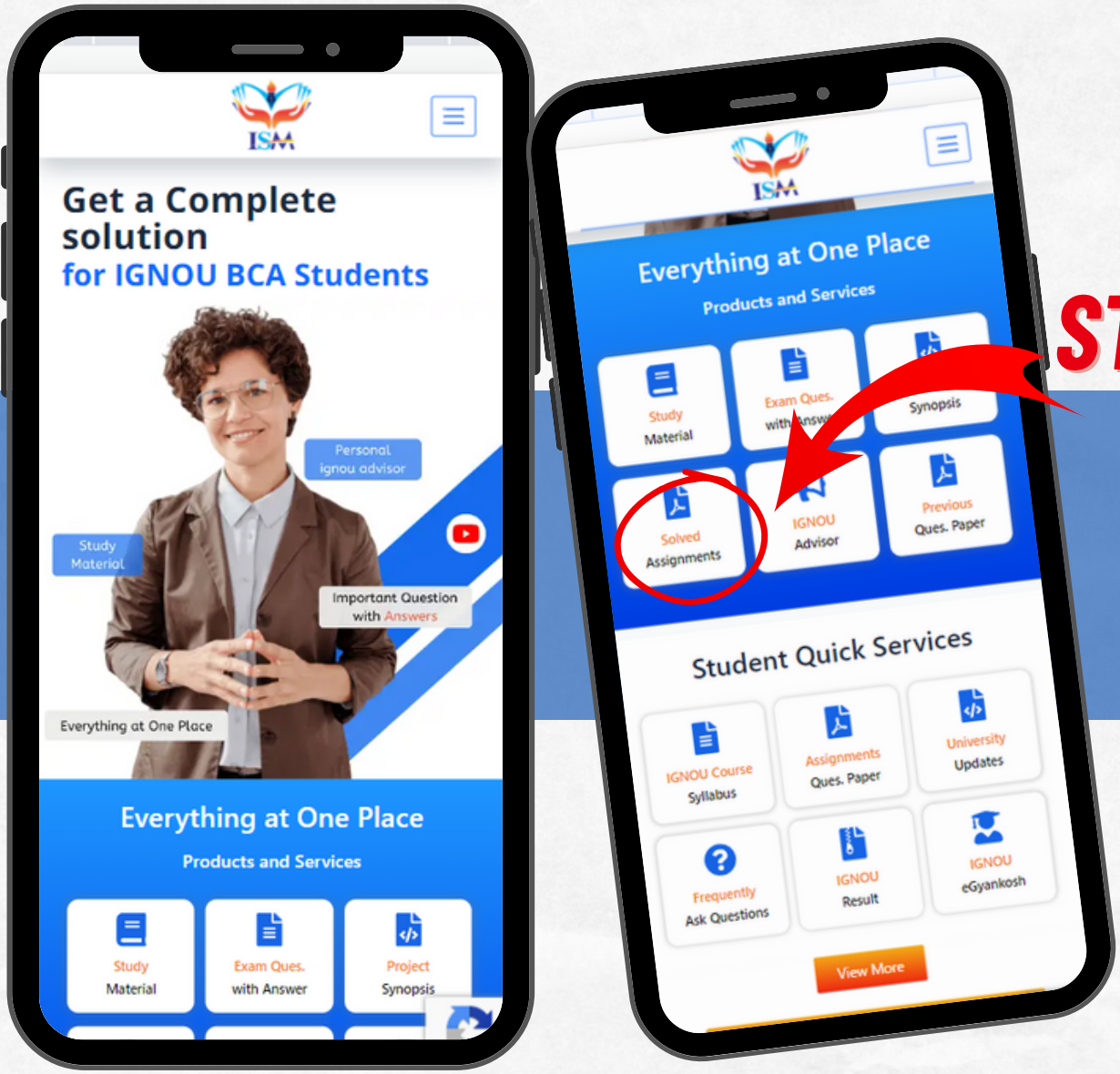
Get a Complete Solution

for IGNOU Students

www.ignoustudymentor.com

STEP 1

www.ignoustudymentor.com



STEP 2

MCS-024
OBJECT-ORIENTED TECHNOLOGIES AND JAVA PROGRAMMING [SEM-4]

Ques.3. What is meant by thread in Java? Explain its use with an example.

Ans. In Java, a thread is the smallest unit of execution within a process. It represents a sequence of instructions that can be scheduled and executed independently by the Java Virtual Machine (JVM). Java's multithreading capabilities allow multiple threads to run concurrently within a single program, enabling better utilization of CPU resources and improved responsiveness.

A thread in Java is a lightweight process that allows a program to run multiple tasks simultaneously. Each thread has its own call stack and registers, which allows it to execute its own code independently of other threads.

Threads can be used to improve the performance of a program by allowing multiple tasks to be executed at the same time. For example, a web browser can use multiple threads to download multiple images at the same time, which can significantly improve the time it takes to load a web page.

Threads can also be used to improve the responsiveness of a program. For example, a chat application can use multiple threads to handle incoming messages from multiple users. This ensures that messages are processed quickly and that users do not have to wait for a response.

Key points about threads in Java:

MCS-024
OBJECT-ORIENTED TECHNOLOGIES AND JAVA PROGRAMMING [SEM-4]

1. Threads in Java are managed by the JVM, and the operating system schedules their execution.
2. Each Java application has at least one thread, known as the main thread.
3. Threads share the same memory space, which allows them to communicate and share data, but this can also lead to synchronization and data race issues.

To create a thread in Java, you can extend the Thread class or implement the Runnable interface. If you extend the Thread class, you will need to override the run() method. This method is where you will put the code that you want the thread to execute.

If you implement the Runnable interface, you will need to create a class that implements the run() method. You can then create a Thread object and pass it an instance of your class.

example of creating and using threads in Java:

```
class MyThread extends Thread {
@Override
public void run() {
 for (int i = 1; i <= 5; i++) {
 System.out.println("Thread " +
Thread.currentThread().getId() + ": " + i);
   }
  }
}
```

MCS-024
OBJECT-ORIENTED TECHNOLOGIES AND JAVA PROGRAMMING [SEM-4]


```
public class Main {
   public static void main(String[] args) {
   MyThread thread1 = new MyThread();
   MyThread thread2 = new MyThread();

   thread1.start(); // Start the first thread
   thread2.start(); // Start the second thread
  }
 }
```

Threads are commonly used to perform tasks concurrently, such as handling user input, running background computations, or managing multiple connections in network applications. However, working with threads also requires careful consideration of synchronization and coordination mechanisms to avoid issues like race conditions and deadlocks.

Once a thread has been started, it will begin executing its code in its own call stack. The main thread will continue to execute its code in its own call stack.

Threads can communicate with each other by sharing data or by using synchronization primitives. For example, two threads can share a data object by using the synchronized keyword. This ensures that only one thread can access the data object at a time.

Threads can also be synchronized using locks. A lock is an object that can be acquired by one thread at a time. This ensures that only one thread can execute a critical section of code at a time

Ques.4. What is AWT? How does it differ from swing ?

Ans. AWT (Abstract Window Toolkit): AWT stands for Abstract Window Toolkit. It is the original user interface framework for Java applications. AWT provides a set of classes and methods that allow you to create and manage graphical user interface (GUI) components, such as windows, buttons, text fields, and more. AWT components are native to the underlying operating system and provide a consistent look and feel across different platforms.

AWT components are heavyweight, meaning they are directly mapped to native components of the operating system. This can lead to better integration with the host platform's user interface but might also result in some inconsistencies across different platforms.

Swing: Swing is a more advanced and flexible GUI framework for Java applications. It was developed as a replacement for AWT and addresses some of the limitations and inconsistencies of AWT. Swing components are often referred to as lightweight components because they are not directly tied to native operating system components. Instead, Swing components are drawn and managed entirely by Java code, allowing for a more consistent look and feel across platforms.

Swing provides a rich set of customizable GUI components and supports various themes and styles. It also offers advanced features like pluggable look-and-feel, double buffering, and customizable painting.

MCS-024
OBJECT-ORIENTED TECHNOLOGIES AND JAVA PROGRAMMING [SEM-4]

Differences between AWT and Swing:

1. Component Type:- AWT: Provides heavyweight components directly mapped to native operating system components.- Swing: Provides lightweight components that are drawn and managed by Java code.

2. Look and Feel:- AWT: Components have a native look and feel, which can vary across different platforms.- Swing: Components can have a consistent look and feel across platforms due to their lightweight nature.

3. Customizability:- AWT: Limited customization options for appearance and behavior.- Swing: Extensive customization options for appearance, behavior, and themes.

4. Performance:- AWT: Generally performs well but might have platform-specific performance issues.- Swing: Might have slightly lower performance compared to AWT due to its lightweight nature, but it's often negligible.

5. Complexity and Features:- AWT: Simpler and more straightforward, with fewer features.- Swing: More feature-rich and complex, offering advanced features and components.

6. Compatibility:- AWT: Tightly integrated with native components, leading to better compatibility with native OS features.- Swing: May have some compatibility issues due to its lightweight nature, but these issues are minimal in most cases

Follow :

ism_ignoustudymentor_

Join Telegram Group :

ism_ignoustudymentor

Youtube Channel

ISM - IGNOU Study MENTOR

Website

www.ignoustudymentor.com

730-339-3555