NAME: SAGNIK DAS **ROLL NO:** 55
ENROLLMENT NO: 12023006015037 **SEC:** A **Semester:** 2nd
DEPARTMENT: M.C.A.

Week 9

1.Write a Java program in which total 4 threads should run. Set different priorities to the thread.

Code:

```
class ThreadDemo extends Thread{
	public void run()
	{
		System.out.println("Inside run method");
	}
	public static void main(String[] args)
	{
		ThreadDemo t1 = new ThreadDemo();
		ThreadDemo t2 = new ThreadDemo();
		ThreadDemo t3 = new ThreadDemo();
		System.out.println("t1 thread priority : "
					+ t1.getPriority());
		System.out.println("t2 thread priority : "
					+ t2.getPriority());
		System.out.println("t3 thread priority : "
					+ t3.getPriority());
		t1.setPriority(2);
		t2.setPriority(5);
		t3.setPriority(8);
		System.out.println("t1 thread priority : "
					+ t1.getPriority());
		    System.out.println("t2 thread priority : "
					+ t2.getPriority());
		System.out.println("t3 thread priority : "
					+ t3.getPriority());
		System.out.println("Currently Executing Thread : "+
Thread.currentThread().getName());
		System.out.println(
			"Main thread priority : "
			+ Thread.currentThread().getPriority());
		Thread.currentThread().setPriority(10);
		System.out.println(
			"Main thread priority : "
			+ Thread.currentThread().getPriority());
	}
}
```

```
t1 thread priority : 5
t2 thread priority : 5
t3 thread priority : 5
t1 thread priority : 2
t2 thread priority : 5
t3 thread priority : 8
Currently Executing Thread : main
Main thread priority : 5
Main thread priority : 10
PS D:\JAVA_Lab_Assignments>
```

2. Write a Java Program to Use Method Level Synchronization.

Code:

```
import java.io.*;
class Line
{
public void getLine()
{
        for (int i = 0; i < 3; i++)
        {
                System.out.println(i);
                try
                {
                        Thread.sleep(400);
                }
                catch (Exception e)
                {
                        System.out.println(e);
                }
        }
}
}
class Train extends Thread
{
Line line;
Train(Line line)
{
        this.line = line;
}
@Override
public void run()
{
        line.getLine();
}
}

class GFG
```

```
	{
	public static void main(String[] args)
	{
		Line obj = new Line();
		Train train1 = new Train(obj);
		Train train2 = new Train(obj);
		train1.start();
		train2.start();
	}
	}
```

```
0
0
1
1
2
2
```

3. Write a Java Program to Use Block Level Synchronization.

Code:

```
import java.io.*;
import java.util.*;
public class Geek
{
	String name = "";
	public int count = 0;

	public void geekName(String geek, List<String> list)
	{
		synchronized(this)
		{
			name = geek;
			count++;
		}
		list.add(geek);
	}
}
class GFG
{
	public static void main (String[] args)
	{
		Geek gk = new Geek();
		List<String> list = new ArrayList<String>();
		gk.geekName("mohit", list);
		System.out.println(gk.name);

	}
}
```

mohit

4. Write a Java Program to Check Whether Define run() Method as Synchronized.

Code:

```
import java.io.*;
import java.util.*;
class Sender {
	public void send(String msg)
	{
		System.out.println("Sending\t" + msg);
		try {
			Thread.sleep(1000);
		}
		catch (Exception e) {
			System.out.println("Thread interrupted.");
		}
		System.out.println("\n" + msg + "Sent");
	}
}
class ThreadedSend extends Thread {
	private String msg;
	Sender sender;
	ThreadedSend(String m, Sender obj)
	{
		msg = m;
		sender = obj;
	}
	public void run()
	{
		synchronized (sender)
		{
			sender.send(msg);
		}
	}
}
class SyncDemo {
	public static void main(String args[])
	{
		Sender send = new Sender();
		ThreadedSend S1 = new ThreadedSend(" Hi ", send);
		ThreadedSend S2 = new ThreadedSend(" Bye ", send);
		S1.start();
		S2.start();
		try {
			S1.join();
			S2.join();
		}
		catch (Exception e) {
			System.out.println("Interrupted");
```

```
                }
        }
}
```

```
Sending     Hi


 Hi Sent

Sending     Bye


 Bye Sent
```

5. Write a Java Program to Solve Producer Consumer Problem Using Synchronization.

Code:

```
import java.util.LinkedList;
public class Threadexample {
        public static void main(String[] args)
                throws InterruptedException
        {
                final PC pc = new PC();
                Thread t1 = new Thread(new Runnable() {
                        @Override
                        public void run()
                        {
                                try {
                                        pc.produce();
                                }
                                catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                        }
                });
                Thread t2 = new Thread(new Runnable() {
                        @Override
                        public void run()
                        {
                                try {
                                        pc.consume();
                                }
                                catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                        }
                });
                t1.start();
                t2.start();
                t1.join();
                t2.join();
```

```
        }
        public static class PC {
                LinkedList<Integer> list = new LinkedList<>();
                int capacity = 2;
                public void produce() throws InterruptedException
                {
                        int value = 0;
                        while (true) {
                                synchronized (this)
                                {
                                while (list.size() == capacity)
                                                wait();

                                System.out.println("Producer produced-"+ value);
                                list.add(value++);
                                notify();
                                Thread.sleep(1000);
                                }
                        }
                }
                public void consume() throws InterruptedException
                {
                                while (true) {
                                synchronized (this)
                                {
                                   while (list.size() == 0)
                                    wait();
                                    int val = list.removeFirst();
                                    System.out.println("Consumer consumed-"+ val);
                                    Notify();
                                    Thread.sleep(1000);
                                }
                        }
                }
        }
}
```

```
Producer produced-0

Producer produced-1

Consumer consumed-0

Consumer consumed-1

Producer produced-2
```

6.Write a Java Program to Show that Method Will be Verified Whether it is Synchronized or Not.

Code:

```
public class SynchronizedMethodVerification {
   public synchronized void synchronizedMethod() {
      System.out.println("This is a synchronized method.");
```

```
    }
    public void nonSynchronizedMethod() {
        System.out.println("This is not a synchronized method.");
    }

    public static void main(String[] args) {
        SynchronizedMethodVerification obj = new SynchronizedMethodVerification();
        try {
            System.out.println("Verifying synchronized method:");
            obj.synchronizedMethod();
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        try {
            System.out.println("Verifying non-synchronized method:");
            obj.nonSynchronizedMethod();
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
Verifying synchronized method:
This is a synchronized method.
Verifying non-synchronized method:
This is not a synchronized method.
```

7. Write a Java Program to Show How Can Class Object be Locked Using Method Level Synchronization.

Code:

```
public class ClassLockDemo {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new MyClass());
        Thread thread2 = new Thread(new MyClass());
        thread1.start();
        thread2.start();
    }
}
class MyClass implements Runnable{
    public static synchronized void synchronizedMethod() {
        System.out.println(Thread.currentThread().getName() + " is executing synchronized
method.");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
        }
        System.out.println(Thread.currentThread().getName() + " finished executing
synchronized method.");
    }
    @Override
    public void run(){
        synchronizedMethod();
    }
}
```

```
Thread-0 is executing synchronized method.
Thread-0 finished executing synchronized method.
Thread-1 is executing synchronized method.
Thread-1 finished executing synchronized method.
```

8. Write a Java Program to Synchronize the Threads Acting on the Same Object. The Synchronized Block in the Program can be Executed by Only One Thread at a Time.

Code:

```
class Counter {
    private int count = 0;
    public void increment() {
        synchronized(this) {
            count++;
            System.out.println(Thread.currentThread().getName() + " increments count to: " +
count);
        }
    }
    public int getCount() {
        return count;
    }
}
class IncrementThread extends Thread {
    private Counter counter;
    public IncrementThread(Counter counter) {
        this.counter = counter;
    }
    public void run() {
        for (int i = 0; i < 5; i++) {
            counter.increment();
            try {
                Thread.sleep(100); // Sleep for some time to simulate other operations
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Counter counter = new Counter();
        IncrementThread thread1 = new IncrementThread(counter);
        IncrementThread thread2 = new IncrementThread(counter);
        thread1.setName("Thread 1");
        thread2.setName("Thread 2");
        thread1.start();
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Final Count: " + counter.getCount());
    }
}
```

```
Thread 1 increments count to: 1
Thread 1 increments count to: 2
Thread 2 increments count to: 3
Thread 1 increments count to: 4
Thread 2 increments count to: 5
Thread 2 increments count to: 6
Thread 2 increments count to: 7
Thread 1 increments count to: 8
Thread 1 increments count to: 9
Thread 2 increments count to: 10
Final Count: 10
```

9. Write a Java Program to Avoid Dead Locks.

Code:

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
public class DeadlockAvoidanceExample {
    private static final Lock lock1 = new ReentrantLock();
    private static final Lock lock2 = new ReentrantLock();
    public static void main(String[] args) {
        Thread thread1 = new Thread(() -> {
            acquireLocks(lock1, lock2);
        });

        Thread thread2 = new Thread(() -> {
            acquireLocks(lock2, lock1);
        });
        thread1.start();
        thread2.start();
    }
    private static void acquireLocks(Lock firstLock, Lock secondLock) {
```

```
        firstLock.lock();
        System.out.println(Thread.currentThread().getName() + " acquired " + firstLock);
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        secondLock.lock();
        System.out.println(Thread.currentThread().getName() + " acquired " + secondLock);
        secondLock.unlock();
        System.out.println(Thread.currentThread().getName() + " released " + secondLock);
        firstLock.unlock();
        System.out.println(Thread.currentThread().getName() + " released " + firstLock);
    }
}
```

```
Thread-0 acquired java.util.concurrent.locks.ReentrantLock@648a2ae1[Locked by thread Thread-0]
Thread-1 acquired java.util.concurrent.locks.ReentrantLock@3aa1c752[Locked by thread Thread-1]
```

10. Write a Java Program to Solve Deadlock Using Thread.

Code:

```
public class DeadlockSolution {
    public static Object lock1 = new Object();
    public static Object lock2 = new Object();
    public static void main(String[] args) {
        Thread thread1 = new Thread(new Thread1());
        Thread thread2 = new Thread(new Thread2());
        thread1.setPriority(Thread.NORM_PRIORITY);
        thread2.setPriority(Thread.MAX_PRIORITY);
        thread1.start();
        thread2.start();
    }
    private static class Thread1 implements Runnable {
        public void run() {
            synchronized (lock1) {
                System.out.println("Thread 1: Holding lock 1...");
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("Thread 1: Waiting for lock 2...");

                synchronized (lock2) {
                    System.out.println("Thread 1: Holding lock 1 & 2...");
                }
            }
        }
    }
    private static class Thread2 implements Runnable {
        public void run() {
```

```
            synchronized (lock2) {
                System.out.println("Thread 2: Holding lock 2...");
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("Thread 2: Waiting for lock 1...");

                synchronized (lock1) {
                    System.out.println("Thread 2: Holding lock 2 & 1...");
                }
            }
        }
    }
}
```

```
Thread 1: Holding lock 1...
Thread 2: Holding lock 2...
Thread 1: Waiting for lock 2...
Thread 2: Waiting for lock 1...
```

11. Write a Java Program to Create a Thread that Implement the Runnable Interface.

Code:

```
class MyRunnable implements Runnable {
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println("Thread running: " + i);
            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted!");
            }
        }
    }
}
public class Main {
    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();
        Thread thread = new Thread(myRunnable);
        thread.start();
        for (int i = 0; i < 5; i++) {
            System.out.println("Main thread running: " + i);
            try {
                Thread.sleep(1500); // Sleep for 1.5 seconds
            } catch (InterruptedException e) {
                System.out.println("Main thread interrupted!");
            }
        }
```

```
    }
}
```

```
Main thread running: 0
Thread running: 0
Main thread running: 1
Thread running: 1
Main thread running: 2
Thread running: 2
Thread running: 3
Main thread running: 3
Thread running: 4
Main thread running: 4
```

12.Write a Java Program to Show the Priority in Threads.

Code:

```
class PriorityDemo implements Runnable {
    private String name;
    public PriorityDemo(String name) {
        this.name = name;
    }
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(name + " is running iteration " + i);
            try {
                Thread.sleep(100); // Sleep for 100 milliseconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        PriorityDemo demo1 = new PriorityDemo("Thread 1");
        PriorityDemo demo2 = new PriorityDemo("Thread 2");
        Thread t1 = new Thread(demo1);
        Thread t2 = new Thread(demo2);
        t1.setPriority(Thread.MIN_PRIORITY);
        t2.setPriority(Thread.MAX_PRIORITY);
        t1.start();
        t2.start();
    }
}
```

```
Thread 2 is running iteration 0
Thread 1 is running iteration 0
Thread 1 is running iteration 1
Thread 2 is running iteration 1
Thread 1 is running iteration 2
Thread 2 is running iteration 2
Thread 1 is running iteration 3
Thread 2 is running iteration 3
Thread 1 is running iteration 4
Thread 2 is running iteration 4
PS D:\JAVA_Lab_Assignments>
```

13.Write a Java Program to Check Priority Level of a Thread.

Code:
```
class PriorityChecker implements Runnable {
    public void run() {
        System.out.println("Thread priority is: " + Thread.currentThread().getPriority());
    }
    public static void main(String[] args) {
        PriorityChecker priorityChecker = new PriorityChecker();
        Thread thread = new Thread(priorityChecker);
        thread.setPriority(Thread.NORM_PRIORITY);
        thread.start();
    }
}
```

```
Thread priority is: 5
PS D:\JAVA_Lab_Assignments>
```

14. Write a Java Program to Set the Priority of a Thread.

Code:
```
class PrioritySetter implements Runnable {
    private String name;
    public PrioritySetter(String name) {
        this.name = name;
    }
    public void run() {
        System.out.println(name + " is running with priority " +
Thread.currentThread().getPriority());
    }
public static void main(String[] args) {
        PrioritySetter lowPriority = new PrioritySetter("Low Priority Thread");
        PrioritySetter highPriority = new PrioritySetter("High Priority Thread");
        Thread lowThread = new Thread(lowPriority);
        Thread highThread = new Thread(highPriority);
        lowThread.setPriority(Thread.MIN_PRIORITY);
        highThread.setPriority(Thread.MAX_PRIORITY);
        lowThread.start();
        highThread.start();
    }
```

```
}
Low Priority Thread is running with priority 1
High Priority Thread is running with priority 10
PS D:\JAVA_Lab_Assignments>
```

15. Write a Java Program to Get the Priorities of Running Threads.

Code:

```
public class ThreadPriorityDemo {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new MyRunnable(), "Thread 1");
        Thread thread2 = new Thread(new MyRunnable(), "Thread 2");
        Thread thread3 = new Thread(new MyRunnable(), "Thread 3");
        thread1.setPriority(Thread.MIN_PRIORITY);
        thread2.setPriority(Thread.NORM_PRIORITY);
        thread3.setPriority(Thread.MAX_PRIORITY);
        thread1.start();
        thread2.start();
        thread3.start();
        try {
            thread1.join();
            thread2.join();
            thread3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    static class MyRunnable implements Runnable {
        @Override
        public void run() {
            System.out.println(Thread.currentThread().getName() + " Priority: " +
Thread.currentThread().getPriority());
        }
    }
}
```

```
Thread 1 Priority: 1
Thread 3 Priority: 10
Thread 2 Priority: 5
PS D:\JAVA_Lab_Assignments>
```

16. Write a Java Program to Access the Priority You Can Use Method With Thread Object.

Code:

```
class MyThread extends Thread {
public void run() {
```

```
System.out.println("Thread Name: " + Thread.currentThread().getName());
System.out.println("Thread Priority: " + Thread.currentThread().getPriority());
}
}
public class Main {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
MyThread thread2 = new MyThread();
MyThread thread3 = new MyThread();
thread1.setPriority(Thread.MIN_PRIORITY); // 1
thread2.setPriority(Thread.NORM_PRIORITY); // 5
thread3.setPriority(Thread.MAX_PRIORITY); // 10
thread1.start();
thread2.start();
thread3.start();
}
}
```

```
Thread Name: Thread-0
Thread Priority: 1
Thread Name: Thread-1
Thread Priority: 5
Thread Name: Thread-2
Thread Priority: 10
```

17. Write a Java Program to Use Join Thread.

Code:

```
class MyThread extends Thread {
public void run() {
for (int i = 1; i <= 5; i++) {
System.out.println(Thread.currentThread().getName() + ": " + i);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println(e);
}
}
}
}
public class Main {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
MyThread thread2 = new MyThread();
MyThread thread3 = new MyThread();
thread1.setName("Thread 1");
thread2.setName("Thread 2");
thread3.setName("Thread 3");
thread1.start();
try {
```

```
thread1.join(); // Wait for thread1 to finish
} catch (InterruptedException e) {
System.out.println(e);
}
thread2.start();
try {
thread2.join(); // Wait for thread2 to finish
} catch (InterruptedException e) {
System.out.println(e);
}
thread3.start();
try {
thread3.join(); // Wait for thread3 to finish
} catch (InterruptedException e) {
System.out.println(e);
}
System.out.println("All threads have finished executing.");
}
}
```

```
Thread 1: 1
Thread 1: 2
Thread 1: 3
Thread 1: 4
Thread 1: 5
Thread 2: 1
Thread 2: 2
Thread 2: 3
Thread 2: 4
Thread 2: 5
Thread 3: 1
Thread 3: 2
Thread 3: 3
Thread 3: 4
Thread 3: 5
All threads have finished executing.
```

18. Write a Java Program Defining Thread By Extending Thread.

Code:

```
class MyThread extends Thread {
public void run() {
for (int i = 1; i <= 5; i++) {
System.out.println(Thread.currentThread().getName() + ": " + i);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println(e);
}
}
}
}
public class Main {
```

```
public static void main(String[] args) {
MyThread thread1 = new MyThread();
MyThread thread2 = new MyThread();
thread1.setName("Thread 1");
thread2.setName("Thread 2");
thread1.start();
thread2.start();
}
}
```

```
Thread 1: 1
Thread 2: 1
Thread 1: 2
Thread 2: 2
Thread 1: 3
Thread 2: 3
Thread 1: 4
Thread 2: 4
Thread 1: 5
Thread 2: 5
```

19. Write a Java Program to Handle IllegalThreadStateException.

Code:

```
class MyThread extends Thread {
public void run() {
try {
System.out.println("Thread is running");
Thread.sleep(2000);
} catch (InterruptedException e) {
System.out.println(e);
}
}
}
public class Main {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
try {
thread.start();
} catch (IllegalThreadStateException e) {
System.out.println("IllegalThreadStateException caught: " + e.getMessage());
}
}
}
```

```
Thread is running
IllegalThreadStateException caught: Thread already started.
```

20.Write a Java Program to Check Whether Static Block will be Used.

Code:
```
public class Main {
static {
System.out.println("Static block is executed.");
}
public static void main(String[] args) {
System.out.println("Main method is executed.");
}
}
```

```
Static block is executed.
Main method is executed.
```

21. Write a Java Program to Show Why Exit Method is Used in Static Method.

Code:
```
public class ExitExample {
public static void main(String[] args) {
System.out.println("Starting the program.");
// Calling a static method to demonstrate the use of System.exit()
performOperation(5);
// This line won't be executed if System.exit() is called within performOperation()
System.out.println("End of the program.");
}
public static void performOperation(int value) {
if (value < 0) {
System.out.println("Invalid value provided. Exiting the program.");
System.exit(1);
} else {
System.out.println("Valid value provided: " + value);
}
}
}
```

```
Starting the program.
Valid value provided: 5
End of the program.
```

22. Write a Java Program to Illustrate Thread Example for setName(string name).

Code:
```
class MyThread extends Thread {
public MyThread(String name) {
super(name); }
public void run() {
System.out.println("Thread " + getName() + " is running.");
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread1 = new MyThread("Thread-A");
MyThread thread2 = new MyThread("Thread-B");
thread1.setName("MyCustomThread1");
thread2.setName("MyCustomThread2");
thread1.start();
thread2.start();
}
}
```

```
Thread MyCustomThread1 is running.
Thread MyCustomThread2 is running.
```

23. Write a Java Program to Illustrate Thread Example for Destroy().

Code:

```
class MyThread extends Thread {
public MyThread(String name) {
super(name);
}
public void run() {
while (!Thread.interrupted()) {
System.out.println("Thread " + getName() + " is running.");
try {
Thread.sleep(1000); // Simulate some work
} catch (InterruptedException e) {
break;
}
}
System.out.println("Thread " + getName() + " has stopped.");
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread = new MyThread("MyThread");
thread.start();
try {
Thread.sleep(5000); // Main thread sleeps for 5 seconds
} catch (InterruptedException e) {
e.printStackTrace();
}
thread.interrupt();
}
}
```

```
Thread MyThread is running.
Thread MyThread is running.
Thread MyThread is running.
Thread MyThread is running.
Thread MyThread is running.
Thread MyThread has stopped.
```

24.Write a Java Program to Illustrate Thread Example for suspend().

Code:

```
class MyThread extends Thread {
private boolean suspended = false;
public void suspendThread() {
suspended = true;
}
public synchronized void resumeThread() {
suspended = false;
notify();
}
public void run() {
while (true) {
synchronized (this) {
while (suspended) {
try {
wait();
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
System.out.println("Thread is running...");
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
try {
Thread.sleep(3000);
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println("Suspending thread...");
thread.suspendThread();
try {
Thread.sleep(3000);
} catch (InterruptedException e) {
e.printStackTrace();
```

```
}
System.out.println("Resuming thread...");
thread.resumeThread();
}
}
```

```
Thread is running...
Thread is running...
Thread is running...
Suspending thread...
Resuming thread...
Thread is running...
Thread is running...
```

25. Write a Java Program to Illustrate Thread Example for currentThread().

Code:

```
class MyThread extends Thread {
public void run() {
Thread currentThread = Thread.currentThread();
System.out.println("Current Thread: " + currentThread.getName());
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
thread1.start();
MyThread thread2 = new MyThread();
thread2.start();
}
}
```

```
Current Thread: Thread-0
Current Thread: Thread-1
```

26. Write a Java Program to Illustrate Thread Example for run().

Code:

```
class MyRunnable implements Runnable {
public void run() {
System.out.println("This is a runnable thread.");
}
}
public class RunnableExample {
public static void main(String[] args) {
MyRunnable myRunnable = new MyRunnable();
Thread thread = new Thread(myRunnable);
thread.start();
}
}
```

```
This is a runnable thread.
```

27.Write a Java Program to Illustrate Thread Example for getThreadGroup().

Code:
```
class MyThread extends Thread {
public void run() {
ThreadGroup threadGroup = Thread.currentThread().getThreadGroup();
System.out.println("Thread Group Name: " + threadGroup.getName());
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
thread1.start();
MyThread thread2 = new MyThread();
thread2.start();
}
}
```

```
Thread Group Name: main
Thread Group Name: main
```

28) Write a Java Program to Illustrate Thread Example for getPriority().

Code:
```
class MyThread extends Thread {
public void run() {
int priority = Thread.currentThread().getPriority();
System.out.println("Thread Priority: " + priority);
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
thread1.start();
thread1.setPriority(Thread.MIN_PRIORITY);
MyThread thread2 = new MyThread();
thread2.start();
thread2.setPriority(Thread.MAX_PRIORITY);
}
}
```

```
Thread Priority: 5
Thread Priority: 10
```

29. Write a Java Program to Illustrate Thread Example for Alive().

Code:
```
class MyThread extends Thread {
public void run() {
System.out.println("Thread is running...");
try {
Thread.sleep(2000); // Simulating some work
} catch (InterruptedException e) {
```

```
e.printStackTrace();
}
System.out.println("Thread is finishing...");
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread = new MyThread();
System.out.println("Thread status before starting: " + thread.isAlive());
thread.start();
System.out.println("Thread status after starting: " + thread.isAlive());
try {
Thread.sleep(3000); // Main thread sleeps for 3 seconds
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println("Thread status after completion: " + thread.isAlive());
}
}
```

```
Thread status before starting: false
Thread is running...
Thread status after starting: true
Thread is finishing...
Thread status after completion: false
```

30. Write a Java Program to Illustrate Thread Example for getName().

Code:

```
class MyThread extends Thread {
public void run() {
System.out.println("Thread is running with name: " + getName());
}
}
public class ThreadExample {
public static void main(String[] args) {
MyThread thread1 = new MyThread();
thread1.setName("Thread-1");
thread1.start();
MyThread thread2 = new MyThread();
thread2.setName("Thread-2");
thread2.start();
}
}
```

```
Thread is running with name: Thread-1
Thread is running with name: Thread-2
```

31. Write a Java Program to Show Interfaces Can be Extended.

Code: interface Shape {
double area();
}
interface ThreeDimensionalShape extends Shape {
double volume();
}
class Circle implements Shape {
private double radius;
public Circle(double radius) {
this.radius = radius;
}
@Override
public double area() {
return Math.PI * radius * radius;
}
}
class Sphere implements ThreeDimensionalShape {
private double radius;
public Sphere(double radius) {
this.radius = radius;
}
@Override
public double area() {
return 4 * Math.PI * radius * radius;
}
@Override
public double volume() {
return (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
}
}
public class Main {
public static void main(String[] args) {
Circle circle = new Circle(5);
System.out.println("Area of Circle: " + circle.area());
Sphere sphere = new Sphere(5);
System.out.println("Area of Sphere: " + sphere.area());
System.out.println("Volume of Sphere: " + sphere.volume());
}
}

```
Area of Circle: 78.53981633974483
Area of Sphere: 314.1592653589793
Volume of Sphere: 523.5987755982989
```

32. Write a Java Program to Check a Thread is Alive or Not.
Code: class MyThread extends Thread {
public void run() {

```
try {
Thread.sleep(2000); // Simulating some task
} catch (InterruptedException e) {
System.out.println(e);
}
}
}
public class ThreadAliveCheck {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
if (thread.isAlive()) {
System.out.println("Thread is alive.");
} else {
System.out.println("Thread is not alive.");
}
try {
Thread.sleep(3000); // Waiting for the thread to finish
} catch (InterruptedException e) {
System.out.println(e);
}
// Check again after the thread has finished
if (thread.isAlive()) {
System.out.println("Thread is still alive.");
} else {
System.out.println("Thread is not alive anymore.");
}
}
}
```

```
Thread is alive.
Thread is not alive anymore.
```

33. Write a Java Program to Get the Name of a Running Thread.

Code:

```
public class CurrentThreadName {
public static void main(String[] args) {
Thread currentThread = Thread.currentThread();
String threadName = currentThread.getName();
System.out.println("Name of the currently running thread: " + threadName);
}
}
```

```
Name of the currently running thread: main
```

34.Write a Java Program to Get the Name of the Thread.

Code:
```
public class ThreadNameExample {
public static void main(String[] args) {
Thread currentThread = Thread.currentThread();
String threadName = currentThread.getName();
System.out.println("Current Thread Name: " + threadName);
}
}
```

```
Current Thread Name: main
```

35. Write a Java Program to Check if a Given run() Method is Overloaded in the Thread Class.

Code:
```
import java.lang.reflect.Method;
public class ThreadRunMethodCheck {
public static void main(String[] args) {
Method[] methods = Thread.class.getDeclaredMethods();
Method runMethod = null;
for (Method method : methods) {
if (method.getName().equals("run")) {
runMethod = method;
break;
}
}
System.out.println("Found run method: " + runMethod);
if (isOverloaded(runMethod, Thread.class)) {
System.out.println("The run method in Thread class is overloaded.");
} else {
System.out.println("The run method in Thread class is not overloaded.");
}
}
private static boolean isOverloaded(Method method, Class<?> clazz) {
Method[] methods = clazz.getDeclaredMethods();
for (Method m : methods) {
if (m.getName().equals("run") && !m.equals(method)) {
return true;
}
```

```
Found run method: public void java.lang.Thread.run()
The run method in Thread class is not overloaded.
```

36.Create 4 threads with priority 1,3,5,7 respectively. Update a counter in each of the threads for 10 ms. Print the final value of count for each thread.

Code:
```
import threading
import time
```

```
class CounterThread(threading.Thread):
def __init__(self, priority):
super().__init__()
self.priority = priority
self.counter = 0
def run(self):
# Set thread priority
self.set_priority(self.priority)
# Update counter for 10 ms
start_time = time.time()
while time.time() - start_time < 0.01:
self.counter += 1
print(f"Thread with priority {self.priority}: Final count = {self.counter}")
def set_priority(self, priority):
""" Set thread priority """
if hasattr(threading, 'priority') and hasattr(threading, 'sched_setscheduler'):
# Linux implementation
# Linux kernel priorities range from 1 (highest) to 99 (lowest)
min_prio = 1
max_prio = 99
if priority < min_prio:
priority = min_prio
elif priority > max_prio:
priority = max_prio
policy = threading.sched_setscheduler(0, threading.SCHED_FIFO, (priority,))
if policy < 0:
print("Error setting thread priority.")
elif hasattr(threading, 'priority') and hasattr(threading, 'SetThreadPriority'):
# Windows implementation
# Windows thread priorities range from 1 (lowest) to 15 (highest)
min_prio = 1
max_prio = 15
if priority < min_prio:
priority = min_prio
elif priority > max_prio:
priority = max_prio
threading.SetThreadPriority(priority)
# Create threads with different priorities
thread1 = CounterThread(1)
thread3 = CounterThread(3)
thread5 = CounterThread(5)
thread7 = CounterThread(7)
# Start threads
thread1.start()
thread3.start()
thread5.start()
thread7.start()
# Wait for threads to finish
thread1.join()
thread3.join()
```

thread5.join()
thread7.join()

37. Write a Java Program to Check Whether Define a Thread Class Without Defining run() Method in the Class.

Code:
```
class MyThread extends Thread {
public void run() {
System.out.println("Thread is running.");
}
}
public class Main {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
}
}
```

```
Thread is running.
```

38.Write a Java Program to Stop a Thread.

Code:
```
class MyThread extends Thread {
public void run() {
try {
while (!Thread.currentThread().isInterrupted()) {
System.out.println("Thread is running...");
Thread.sleep(1000); // Simulate some work
}
} catch (InterruptedException e) {
System.out.println("Thread interrupted. Exiting gracefully...");
}
}
}
public class Main {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
try {
Thread.sleep(5000);
} catch (InterruptedException e) {
e.printStackTrace();
}
thread.interrupt();
}
}
```

```
Thread is running...
Thread is running...
Thread is running...
Thread is running...
Thread is running...
Thread interrupted. Exiting gracefully...
```

39.Write a Java Program to Suspend a Thread for a While.

Code:
```
class MyThread extends Thread {
public void run() {
System.out.println("Thread is running...");
try {
Thread.sleep(3000);
} catch (InterruptedException e) {
System.out.println("Thread interrupted while sleeping.");
}
System.out.println("Thread resumes after suspension.");
}
}
public class Main {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
}
}
```

```
Thread is running...
Thread resumes after suspension.
PS D:\JAVA_Lab_Assignments>
```

40.Write a Java Program to Check a Thread has Stopped or Not.

Code:
```
class MyThread extends Thread {
public void run() {
try {
System.out.println("Thread is running...");
Thread.sleep(3000);
} catch (InterruptedException e) {
System.out.println("Thread interrupted while sleeping.");
}
}
}
public class Main {
public static void main(String[] args) {
MyThread thread = new MyThread();
thread.start();
while (thread.isAlive()) {
System.out.println("Thread is still running...");
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
System.out.println("Thread has stopped.");
}
}
```

```
Thread is still running...
Thread is running...
Thread is still running...
Thread is still running...
Thread has stopped.
PS D:\JAVA_Lab_Assignments>
```

NAME: SAGNIK DAS **ROLL NO:** 55
ENROLLMENT NO: 12023006015037 **SEC:** A **Semester:** 2nd
DEPARTMENT: M.C.A.

Week 10

1. Design a Java applet that will blink "Hello Applet" message in the client area and play a musical sound in the background with a background picture in client area.

Code:
```
import java.applet.Applet;
import java.awt.*;
public class BlinkingApplet extends Applet implements Runnable {
private String message = "Hello Applet";
private boolean blink = true;
private Image backgroundImage;
private AudioClip audioClip;
public void init() {
backgroundImage = getImage(getDocumentBase(), "background.jpg");
audioClip = getAudioClip(getDocumentBase(), "background_music.wav");
audioClip.loop();
}
public void start() {
Thread t = new Thread(this);
t.start();
}
public void paint(Graphics g) {
g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
g.setFont(new Font("Arial", Font.BOLD, 20));
g.setColor(Color.RED);
if (blink) {
g.drawString(message, 50, 50);
}
}
public void run() {
while (true) {
blink = !blink;
repaint();
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
```

```
e.printStackTrace();
}
}
}
}
```

2. Design an applet that will display a text as scrolling marquee. The text can be changed by setting different "PARAMS" value.

Code:

```
import java.applet.Applet;
import java.awt.*;
public class ScrollingMarquee extends Applet implements Runnable {
    private String message = "Welcome to Scrolling Marquee!";
    private int xCoordinate = 0;
    private int yCoordinate = 20;
    private int speed = 2; // Adjust scrolling speed
    private Thread thread;
    public void init() {
        String param = getParameter("text");
        if (param != null && !param.isEmpty()) {
            message = param;
        }
    }
    public void start() {
        thread = new Thread(this);
        thread.start();
    }
    public void stop() {
        thread.interrupt();
        thread = null;
    }
    public void run() {
        while (true) {
            xCoordinate -= speed;
            if (xCoordinate < -getWidth()) {
                xCoordinate = getWidth();
            }
            repaint();
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                break;
            }
        }
    }
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());
        g.setFont(new Font("Arial", Font.BOLD, 16));
        g.setColor(Color.BLUE);
        g.drawString(message, xCoordinate, yCoordinate);
```

```
    }
}
```

3.Design a Java applet that displays various shapes like circle, rectangle etc.

Code:

```
import java.applet.Applet;
import java.awt.*;
public class ShapeDrawer extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.drawRect(50, 50, 100, 80);
        g.setColor(Color.BLUE);
        g.fillRect(200, 50, 100, 80);
        g.setColor(Color.GREEN);
        g.drawOval(50, 200, 100, 100);
        g.fillOval(200, 200, 100, 100);
        g.setColor(Color.ORANGE);
        int[] xPoints = {350, 400, 300};
        int[] yPoints = {200, 300, 300};
        g.drawPolygon(xPoints, yPoints, 3);
    }
}
```

4.Design an applet to create digital clock using thread. The clock shows system hh:mm:ss and date.

Code:

```
import javax.swing.*;
import java.awt.*;
import java.util.Calendar;
public class DigitalClock extends JApplet {
    private JLabel timeLabel;
    private JLabel dateLabel;
    @Override
    public void init() {
        SwingUtilities.invokeLater(() -> {
            setLayout(new FlowLayout());
            timeLabel = new JLabel();
            dateLabel = new JLabel();
            add(timeLabel);
            add(dateLabel);
            new TimeThread().start();
        });
    }
    class TimeThread extends Thread {
        @Override
        public void run() {
            try {
                while (true) {
                    Calendar calendar = Calendar.getInstance();
                    int hour = calendar.get(Calendar.HOUR_OF_DAY);
                    int minute = calendar.get(Calendar.MINUTE);
                    int second = calendar.get(Calendar.SECOND);
```

```
                int year = calendar.get(Calendar.YEAR);
                int month = calendar.get(Calendar.MONTH) + 1;
                int day = calendar.get(Calendar.DAY_OF_MONTH);
                String time = String.format("%02d:%02d:%02d", hour, minute, second);
                String date = String.format("%02d/%02d/%d", day, month, year);
                SwingUtilities.invokeLater(() -> {
                    timeLabel.setText("Time: " + time);
                    dateLabel.setText("Date: " + date);
                });
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

5.Write a applet to draw the following shapes: • Rectangle with rounded corner • Square inside a circle.

Code:

```
import javax.swing.*;
import java.awt.*;
public class ShapeDrawingApplet extends JApplet {
    @Override
    public void init() {
        setContentPane(new DrawingPanel());
    }
    class DrawingPanel extends JPanel {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g;
            int rectWidth = 200;
            int rectHeight = 100;
            int arcWidth = 30;
            int arcHeight = 30;
            int rectX = (getWidth() - rectWidth) / 2;
            int rectY = 50;
            g2d.setColor(Color.BLUE);
            g2d.fillRoundRect(rectX, rectY, rectWidth, rectHeight, arcWidth, arcHeight);
            int circleDiameter = 150;
            int circleX = (getWidth() - circleDiameter) / 2;
            int circleY = 200;
            g2d.setColor(Color.RED);
            g2d.fillOval(circleX, circleY, circleDiameter, circleDiameter);

            int squareSize = 100;
            int squareX = circleX + (circleDiameter - squareSize) / 2;
            int squareY = circleY + (circleDiameter - squareSize) / 2;
```

```
            g2d.setColor(Color.GREEN);
            g2d.fillRect(squareX, squareY, squareSize, squareSize);
        }
    }
}
```

6.Write a Java Program to Create Two Lables and Two Text Fields for Entering Name and Passwords. The Password Typed by the User in the Text Field is Hidden.

Code:
```
import javax.swing.*;
import java.awt.*;
public class LoginPanel extends JFrame {
    public LoginPanel() {
        setTitle("Login");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(2, 2));
        JLabel nameLabel = new JLabel("Name:");
        JTextField nameField = new JTextField(20);
        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordField = new JPasswordField(20);
        panel.add(nameLabel);
        panel.add(nameField);
        panel.add(passwordLabel);
        panel.add(passwordField);
        add(panel);
        setVisible(true);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(LoginPanel::new);
    }
}
```

7.Write a Java Program to Display Text in the Frame by Overriding PaintComponent() Method of JPanel Class.

Code:
```
import javax.swing.*;
import java.awt.*;
class TextPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.setColor(Color.BLUE);
        g.drawString("Hello, World!", 50, 50);
    }
}
public class TextFrame extends JFrame {
    public TextFrame() {
        setTitle("Text Display Frame");
```

```
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        TextPanel panel = new TextPanel();
        add(panel);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            TextFrame frame = new TextFrame();
            frame.setVisible(true);
        });
    }
}
```

8. Write a Java Program to Display Some Text in the Frame with the Help of a Label.

import javax.swing.*;

Code:

```
public class TextFrame extends JFrame {
    public TextFrame() {
        setTitle("Text Display Frame");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        JLabel label = new JLabel("Hello, World!");
        add(label);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            TextFrame frame = new TextFrame();
            frame.setVisible(true);
        });
    }
}
```

9. Write a Java Program to Create a Text Area and Display the Mouse Event When the Button on the Mouse is Clicked, When the Mouse is Moved etc is Done by the User.

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MouseEventDemo extends JFrame implements MouseListener,
MouseMotionListener {
    JTextArea textArea;
    public MouseEventDemo() {
        setTitle("Mouse Event Demo");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        textArea = new JTextArea();
        textArea.addMouseListener(this);
        textArea.addMouseMotionListener(this);
```

```
        add(textArea);
    }
    public void mouseClicked(MouseEvent e) {
        textArea.append("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")\n");
    }
    public void mousePressed(MouseEvent e) {
    }
    public void mouseReleased(MouseEvent e) {
    }
    public void mouseEntered(MouseEvent e) {
    }
    public void mouseExited(MouseEvent e) {
    }
    public void mouseDragged(MouseEvent e) {
    }
    public void mouseMoved(MouseEvent e) {
        textArea.append("Mouse Moved to (" + e.getX() + ", " + e.getY() + ")\n");
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            MouseEventDemo frame = new MouseEventDemo();
            frame.setVisible(true);
        });
    }
}
```

10. Write a Java Program to Create a Banner Using Applet

Code:

```
import java.applet.Applet;
import java.awt.*;
public class BannerApplet extends Applet implements Runnable {
    String message = "Welcome to our website!"; // Message to display in the banner
    Thread t;
    boolean stopFlag;
    public void init() {
        setBackground(Color.black);
        setForeground(Color.white);
    }
    public void start() {
        t = new Thread(this);
        stopFlag = false;
        t.start();
    }
    public void run() {
        for (;;) {
            try {
                repaint();
                Thread.sleep(250);
                if (stopFlag)
                    break;
```

```
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted");
            }
        }
    }
    public void stop() {
        stopFlag = true;
        t = null;
    }
    public void paint(Graphics g) {
        char ch;
        ch = message.charAt(0);
        message = message.substring(1, message.length());
        message += ch;
        g.drawString(message, 50, 30);
    }
}
```

11. Write a Java Program to Display Clock Using Applet.

Code:

```
import java.applet.*;
import java.awt.*;
import java.util.*;
public class ClockApplet extends Applet implements Runnable {
    Thread t = null; // Thread that will keep the clock running
    int hours = 0, minutes = 0, seconds = 0; // Time variables
    public void start() {
        if (t == null) {
            t = new Thread(this);
            t.start();
        }
    }
    public void run() {
        try {
            while (true) {
                Calendar cal = Calendar.getInstance();
                hours = cal.get(Calendar.HOUR_OF_DAY);
                if (hours > 12)
                    hours -= 12;
                minutes = cal.get(Calendar.MINUTE);
                seconds = cal.get(Calendar.SECOND);
                repaint();
                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void paint(Graphics g) {
        g.setColor(Color.black);
```

```
        g.drawString(hours + ":" + minutes + ":" + seconds, 20, 20);
    }
}
```

12. Write a Java Program to Create Different Shapes Using Applet.

Code:

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.red);
        g.fillRect(10, 10, 100, 50);
        g.setColor(Color.blue);
        g.fillOval(150, 10, 100, 50);
        g.setColor(Color.green);
        g.fillRoundRect(290, 10, 100, 50, 20, 20);
        int[] xPoints = {450, 500, 550};
        int[] yPoints = {10, 60, 10};
        g.setColor(Color.orange);
        g.fillPolygon(xPoints, yPoints, 3);
        g.setColor(Color.black);
        g.drawLine(10, 120, 550, 120);
    }
}
```

13. Write a Java Program to Fill Colors in Shapes Using Applet.

Code:

```
import java.awt.*;
import java.applet.*;
public class FillShapes extends Applet {
    public void paint(Graphics g) {
        setBackground(Color.white);
        g.setColor(Color.red);
        g.fillRect(10, 10, 100, 50);
        g.setColor(Color.blue);
        g.fillOval(150, 10, 100, 50);
        g.setColor(Color.green);
        g.fillRoundRect(290, 10, 100, 50, 20, 20);
        int[] xPoints = {450, 500, 550};
        int[] yPoints = {10, 60, 10};
        g.setColor(Color.orange);
        g.fillPolygon(xPoints, yPoints, 3);
        g.setColor(Color.magenta);
        g.fillArc(10, 120, 100, 100, 45, 270);
    }
}
```

14. Write a Java Program to go to a Link using Applet.

Code:

```
import java.applet.Applet;
import java.awt.*;
import java.net.*;
public class LinkApplet extends Applet {
    private String linkURL = "http://www.example.com";
```

```
    private String linkLabel = "Click here to visit Example.com";
    public void init() {
        setBackground(Color.white);
    }
    public void paint(Graphics g) {
        g.setFont(new Font("Arial", Font.BOLD, 12));
        g.setColor(Color.blue);
        g.drawString(linkLabel, 20, 20);
    }
    public boolean action(Event event, Object obj) {
        if (event.target instanceof Label) {
            try {
                getAppletContext().showDocument(new URL(linkURL), "_blank");
            } catch (MalformedURLException e) {
                e.printStackTrace();
            }
            return true;
        }
        return false;
    }
}
```

15. Write a Java Program to Create an Event Listener in Applet.

Code:

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.event.*;
public class EventListenerApplet extends Applet {
    public void init() {
        setBackground(Color.WHITE);
        addMouseListener(new CustomMouseListener());
    }
    class CustomMouseListener extends MouseAdapter {
        public void mouseClicked(MouseEvent e) {
            setBackground(Color.RED);
        }
    }
}
```

16. Write a Java Program to Display Image Using Applet.

Code:

```
import java.applet.Applet;
import java.awt.*;
public class DisplayImage extends Applet {
Image img;
public void init() {
img = getImage(getDocumentBase(), "image.jpg");
}
public void paint(Graphics g) {
g.drawImage(img, 0, 0, this); } }
```