|  |  |  |
| --- | --- | --- |
| **MODULE 3** | **COMPUTER PROGRAMMING II** | |
|  |  | |
| **TIME ALLOTMENT** | **1 week** | |
| **INSTRUCTOR** | **Javier, Rodolfo jr. A** | |
|  |  | |
| **FACEBOOK** | **https://www.facebook.com/19javier93** | |
| **EMAIL** | **javierrodolfojr1993@gmail.com** | |
| **CELL NUMBER** | **09504046956** | |
| **I OVERVIEW** | | |
| Computer Programming II focuses on software development using the java programming language and implementing the Object Oriented Programming concepts. This course is designed to advance the knowledge of the students in terms of developing object oriented software using the java language and architecture. This will also include the use of API’s, libraries and repositories to further enhance and improve the functionality and development of the software. | | |
| **II TARGETED COURSE LEARNING OUTCOME** | | |
| CLO4. The students should be able to know the important used of Threading and Multithreading in java programming language. | | |
| **III TARGETED TOPIC LEARNING OUTCOME** | | |
| 1. The students will be able to understanding of fundamental concepts of threads and multitreading in java programming. 2. The students can be able to understanding the typical implementation strategies of threads and multitreading in java programming. 3. The students should be able to develop an application that applied threading and multitreading concept. | | |

Content:

**INTRODUCTION TO THREAD**

• Process and Thread are two basic units of Java program execution.

• Process: A process is a self contained execution environment and it can be seen as a program or

application.

• Thread: It can be called lightweight process

• Thread requires less resources to create and exists in the process

• Thread shares the process resources



**MULTITHREADING**

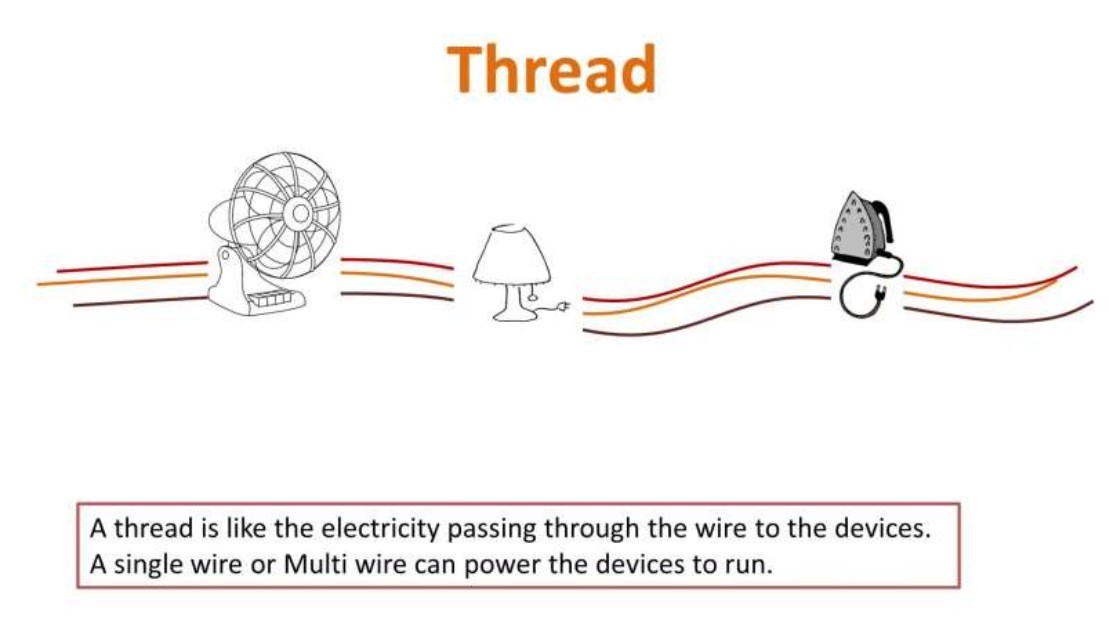
• Multithreading in java is a process of executing multiple processes simultaneously

• A program is divided into two or moresubprograms, which can be implemented at the same time in

parallel.

• Multiprocessing and multithreading, both are used to achieve multitasking.

• Java Multithreading is mostly used in games, animation etc.



**ADVANTAGE**

:



It doesn't block the user



can perform many operations together so it saves time.



Threads are independent so it doesn't affect other threads

**CREATING THREAD**

• Threads are implemented in the form of objects.

• The run() and start() are two inbuilt methods which helps to thread implementation

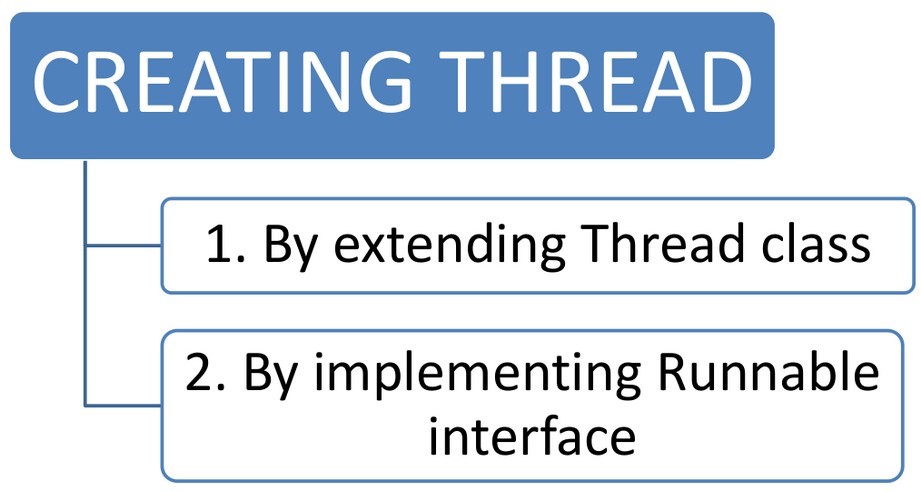
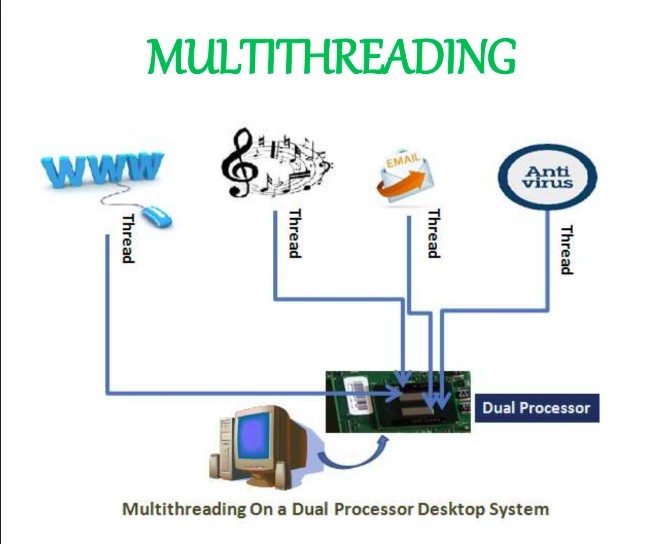
• The run() method is the heart and soul of any thread

– It makes up the entire body of a thread

• The run() method can be initiating with the help of start() method.

**1**

**. By Extending Thread class**



**2**

**. By implementing Runnable interface**



Define a class that implements Runnable interface.



The Runnable interface has only one method, run(), that is to be defined in the method

with the code to be executed by the thread.



**LIFE cycle of a thread**

• During the life time of a thread, there are many states it can enter.

• They include:

1.

Newborn state

2.

Runnable state

3.

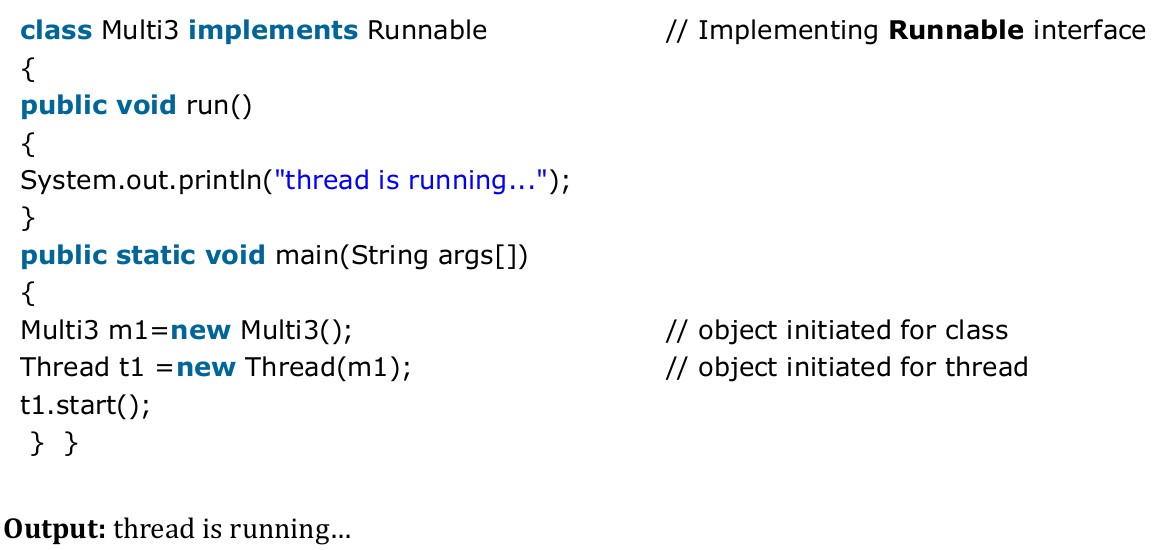
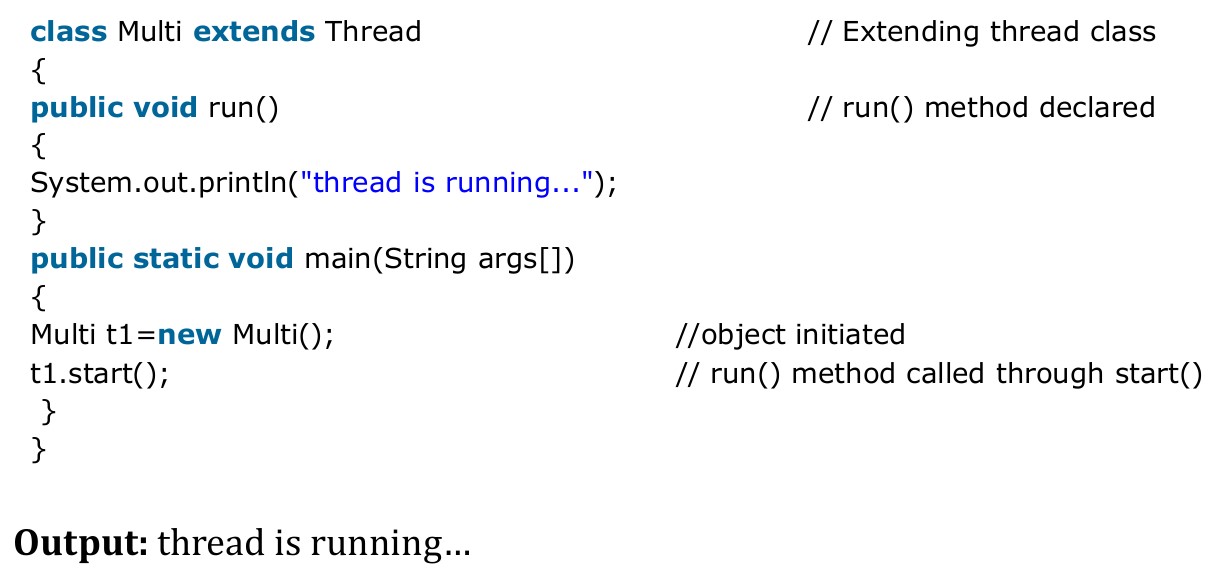
Running state

4.

Blocked state

5.

Dead state



**-**

**Newborn State**

**:**



The thread is born and is said to be in newborn state.



The thread is not yet scheduled for running.



At this state, we can do only one of the following:

• Schedule it for running using start() method.

• Kill it using stop() method.

**-**

**Runnable State**

**:**



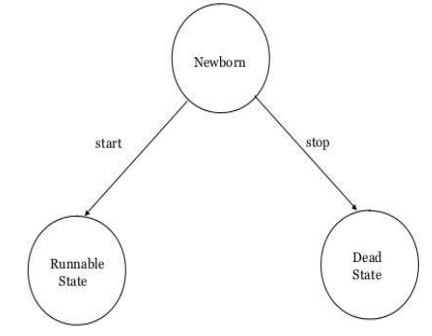
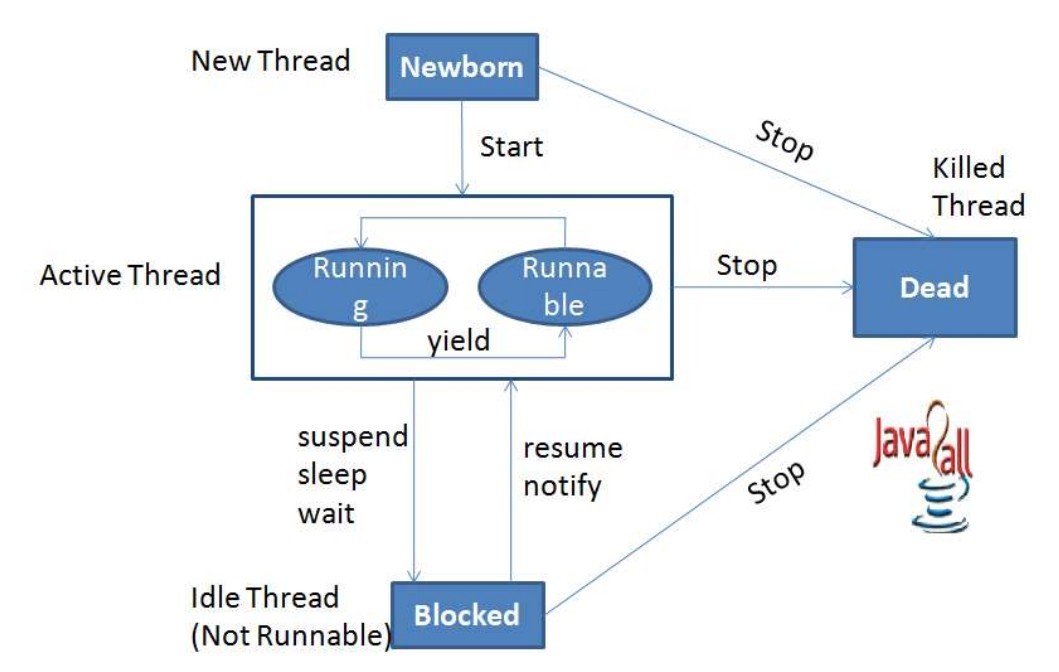
The thread is ready for execution



Waiting for the availability of the processor.



The thread has joined the queue



**-**

**Running State**

**:**

• Thread is executing

• The processor has given its time to the thread for its execution.

• The thread runs until it gives up control on its own or taken over by other threads.

-

**Blocked State:**

**•**

A thread is said to be blocked

• It is prevented to entering into the runnable and the running state.

• This happens when the thread is suspended, sleeping, or waiting in order to satisfy certain

requirements.

• A blocked thread is considered "not runnable" but not dead and therefore fully qualified to run

again.

• This state is achieved when we Invoke suspend() or sleep() or wait() methods.

-

**Dead State:**

• Every thread has a life cycle.

• A running thread ends its life when it has completed executing its run( ) method. It is a natural death.

• A thread can be killed in born, or in running, or even in "not runnable" (blocked) condition.

• It is called premature death.

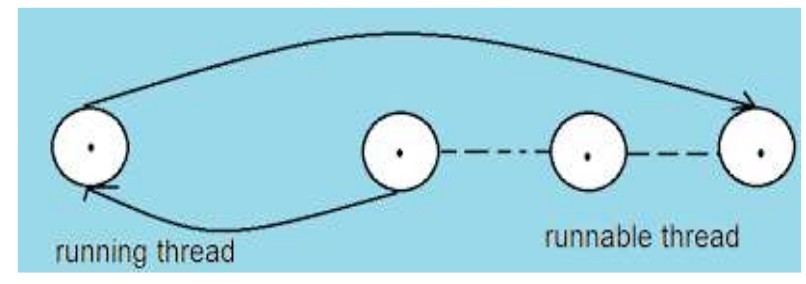
• This state is achieved when we invoke stop() method or the thread completes it execution.



**Thread methods**

**•**

Thread is a class found in java.lang package.



**-**

**Stopping a thread**

**:**

• To stop a thread from running further, we may do so by calling its stop() method.

• This causes a thread to stop immediately and move it to its dead state.

• It forces the thread to stop abruptly before its completion

• It causes premature death.

• To stop a thread we use the following syntax: thread.stop();

**-**

**Blocking a Thread**

**:**

• A thread can also be temporarily suspended or blocked from entering into the runnable and

subsequently running state,

1.

**sleep(t)**

// blocked for ‘t’ milliseconds

2.

**suspend()**

// blocked until resume() method is invoked

3.

**wait()**

// blocked until notify () is invoked

**Thread priority**

**-**

**•**

Each thread is assigned a priority, which affects the order in which it is scheduled for running.

• Java permits us to set the priority of a thread using the setPriority() method as follows:

**ThreadName.setPriority(int Number);**

• The intNumber is an integer value to which the thread's priority is set. The Thread class defines

several priority constants:

1.

public static int

**MIN\_PRIORITY**

= 1

2.

public static int

**NORM\_PRIORITY**

= 5

3.

public static int

**MAX\_PRIORITY**

=

10

• The default setting is NORM\_PRIORITY. Most user-level processes should use

NORM\_PRIORITY.

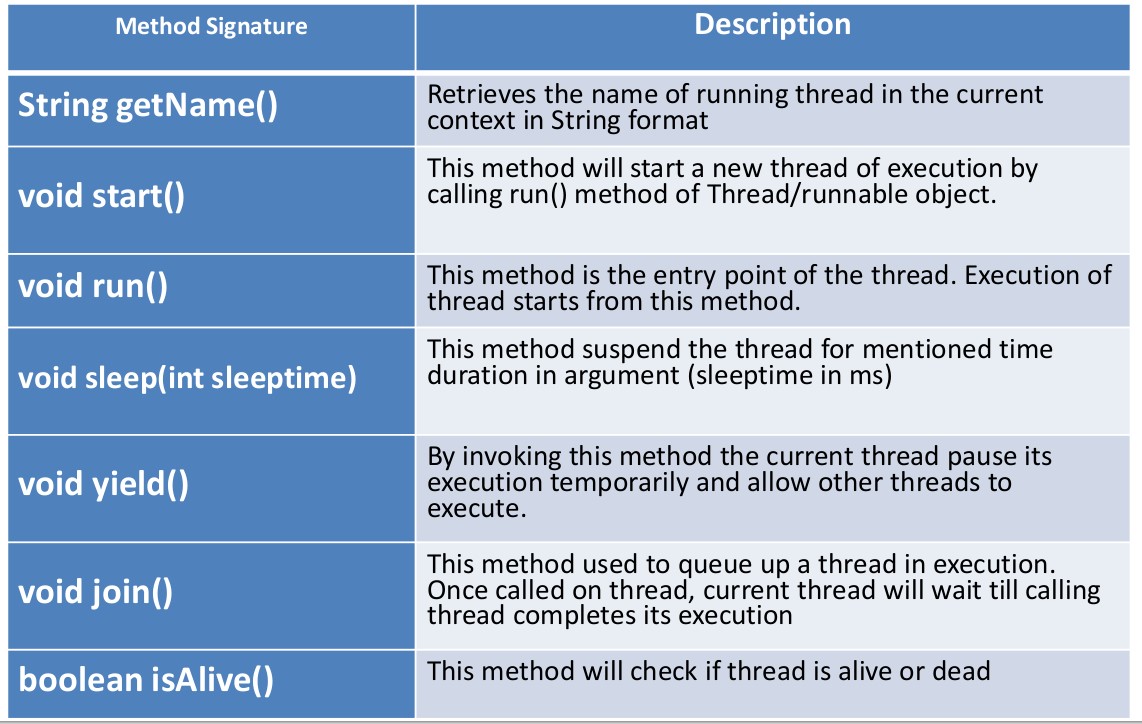
**-**

**Java synchronization**

• Generally threads use their own data and methods provided inside their run() methods.

• But if we wish to use data and methods outside the thread’s run() method, they may compete

for the same resources and may lead to serious problems.



• Java enables us to overcome this problem using a technique known as

**Synchronization**

.

**For ex**

.:

One thread may try to read a record from a file while another is still writing to the

same file.

• When the method declared as synchronized, Java creates a "monitor" and hands it over to the

thread that calls the method first time.

**-**

**deadlock**

• Deadlock describes a situation where two or more threads are blocked forever, waiting for

each other.

• when two or more threads are waiting to gain control on a resource.

For example, assume that the thread A must access Method1 before it can release Method2, but

the thread B cannot release Method1 until it gets holds of Method2.

\* Example Application:

Lets create Stopwatch using threading.

Step 1, Create a project in java netbeans name it

**threadstopwatch.**

Step 2, Create Jframe inside

**threadstopwatch**

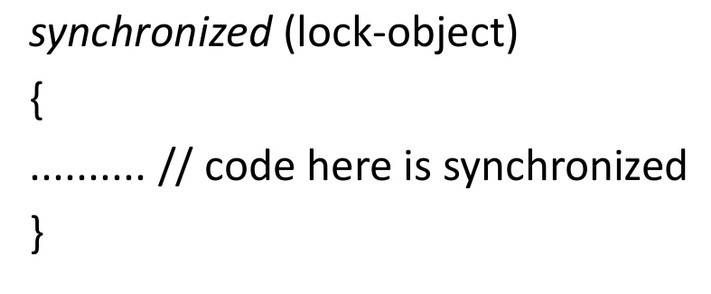
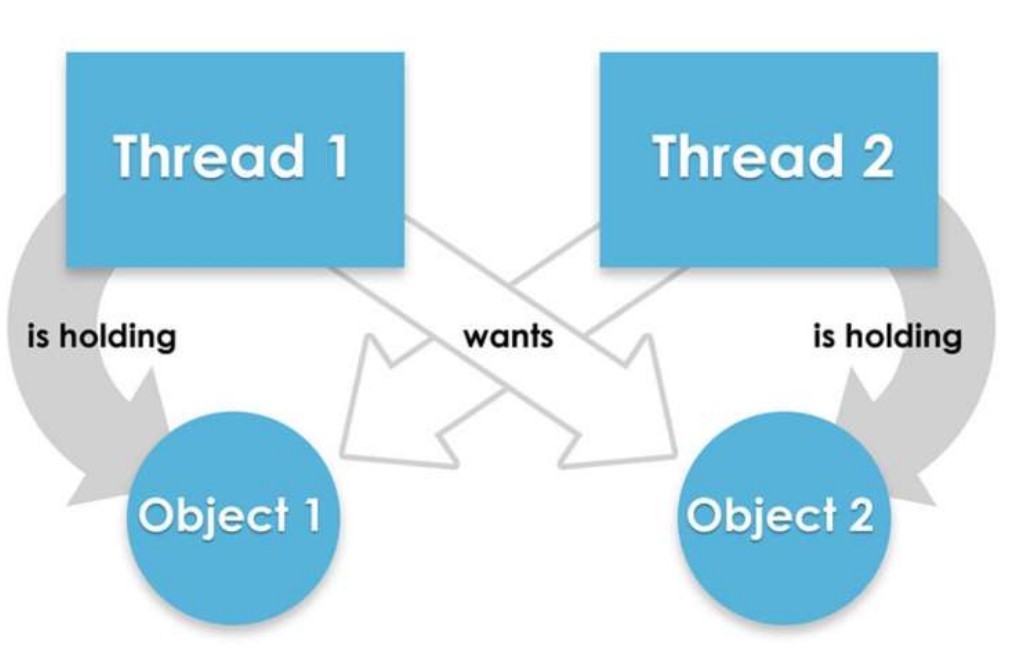
package

and name it to

**stopwatch**

.

Step 3, Follow the design in the image below, name variable name given.

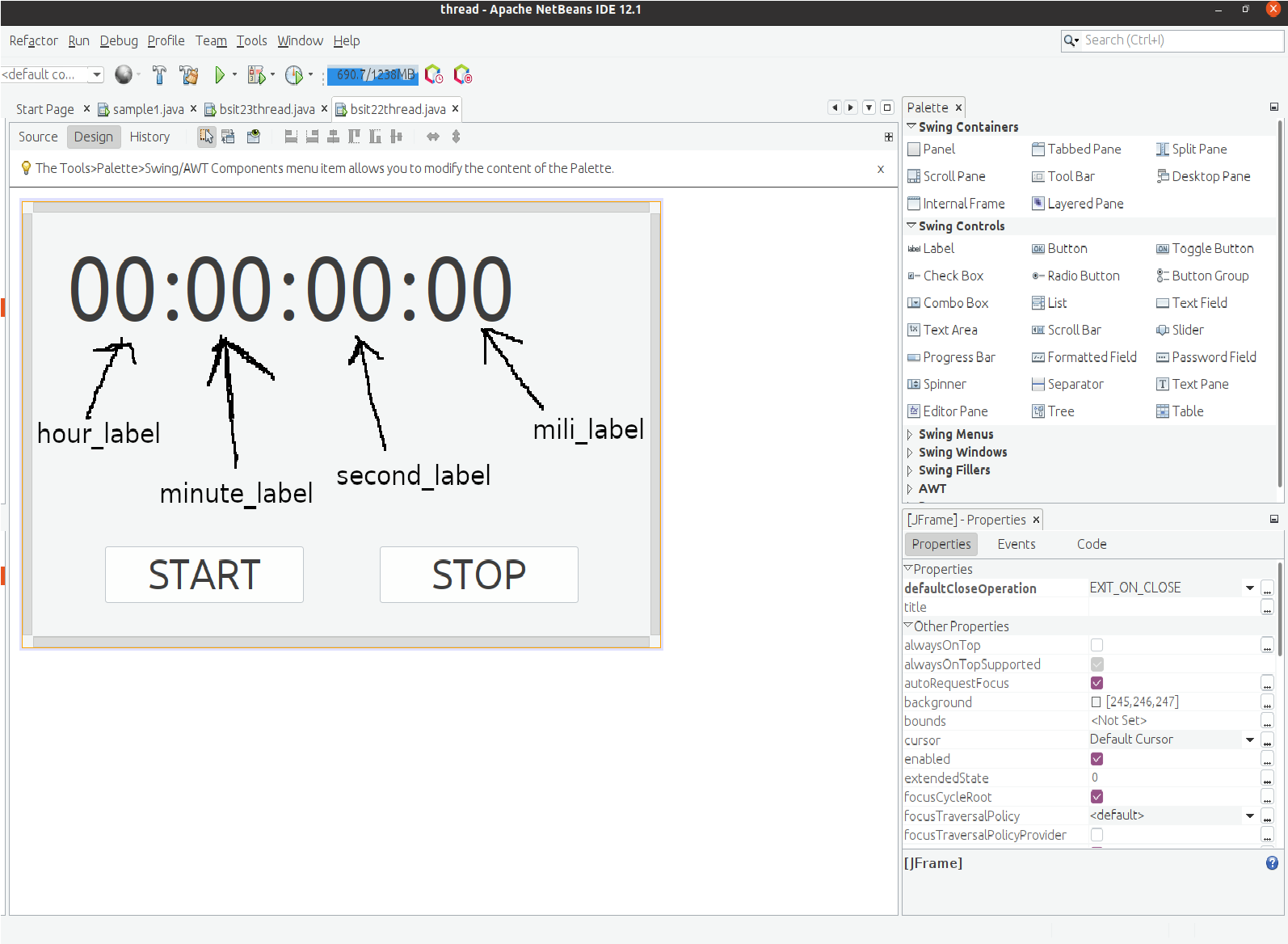
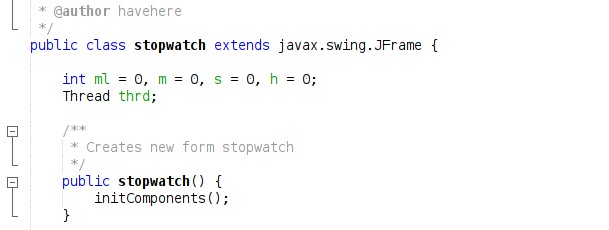
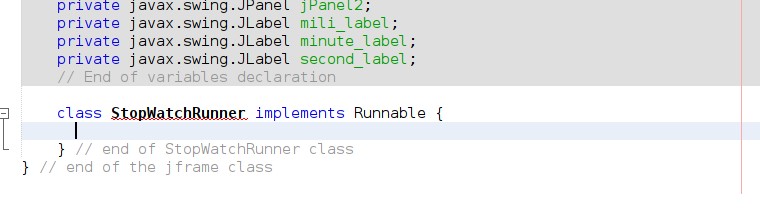


Step 4, declare integer variable name and initialize it globally. See the image below

Step 5, create class name it

**StopWatchRunner**

and implements Runnable



Step 6, click the error bulb at the left line of the class and click implement all abstract.

Step 7, change the code inside the run() method, just copy the code in the image below.

Step 8, double click the

**START**

button on the design and add this code.



|  |
| --- |
| Step 9, double click the **STOP** button on the design and add this code.    Step 10, run the program and this will be the output. |
| **IV ASSESSMENT** |
| **Skills test**: Write a (1) running java program example that using a thread, except stopwatch.  Note: Add a folder on your repository and name it into **module3\_assessment,** put your activity files and commit it on github. |
| **V TEACHING-LEARNING ACTIVITIES** |
| Note: Add a folder and name it into **module3\_tla** (tla means Teaching Learning Activities) before you commit your answer/document.  A **ENGAGE : Misconception ,** Commit your answer/document on your remote repository that collaborated to your instructor github account.  **Misconception Check**  Explain the advantages and disadvantages of treading and multithreading in java programming  **ANSWER:**  Advantages:  Almost every modern processor has multiple cores. You can use them to speed up processing.  Many programs are best expressed as a collection of sequential, communicating processes.  Java implements monitors and conditional variables as part of the base language.  Disadvantages:  Mysterious bugs caused by race conditions  Monitors and the synchronized keyword incur a considerable overhead meaning that plain performance goes down. You have to be able to convince yourself that the advantage of parallelisation outweighs the performance hit to sequential execution.  Java implements monitors and conditional variables. These don’t really isolate processes and data in the way provided by some programming paradigms. The programmer has to be aware of many special cases.  . |
|  |

|  |
| --- |
|  |
| B **EXPLORE : Creating program that can do multithreading**  1. Write a program that can run multiple thread at the same time, and can stop all thread by one click execution. |
| C **EXPLAIN : Reading and Understanding**  To understand the threading process, and how the multithreading work in a flow chart this was the example on the image below. |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **D EVALUATE**  **Self Assessment.**  Kindly check (✔) the box of your answer for each question. In this way, we will be able to assess how much we have learned and what are the things that needs to be improved. | | | |
| **Questions** | **YES** | **NO** | **MAYB E** |
| **1. Did I work hard on this module?** | ✔ |  |  |
| **2. Did I understand what my teacher asked me to do?** | ✔ |  |  |
| **3. Did I spend enough time to finish answering this module?** | ✔ |  |  |
| **4. Did I make good use of available resources?** | ✔ |  |  |
| **5. Did I check/ review my work for possible errors?** | ✔ |  |  |
| **6. Did I learn something in this module?** | ✔ |  |  |
| **7. Did I ask questions if I needed help?** | ✔ |  |  |
| **8. Did I read the instructions carefully?** | ✔ |  |  |
| **9. Did I set high standards for myself?** | ✔ |  |  |
| **10. Did I meet the success criteria?** |  |  | ✔ |