LAB EXERCISE

Multithreading

BITP 3123 DISTRIBUTED APPLICATION DEVELOPMENT

PREPARED BY EMALIANA KASMURI, FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

Table of Contents

Learning Outcomes 1

Exercise 1: Developing Simple Multithreading Application using extends Thread 1

- Task 1: Prepare Development 1
- Task 2: Define Task Class 1
- Task 3: Define Thread Operation 1
- Task 4: Execute Multithreads Application 1

Exercise 2: Developing Simple Multithreading Application using extends Thread 2

- Task 1: Prepare Development 2
- Task 2: Define Task Class 2
- Task 3: Define Thread Operation 2
- Task 4: Execute Multithreads Application 2

Exercise 3: Keeping Track Multiple Threads 2

- Task 1: Name the threads 2
- Task 2: Display the thread's name 3
- Task 4: Execute Multithreads Application 3

Exercise 4: Keeping Track Multiple Threads 3

- Task 1: Name the threads 3
- Task 2: Display the thread's name 3
- Task 4: Execute Multithreads Application 3

Exercise 5: Developing Multithread Application on Different Thread Tasks 4

- Task 1: Create new multithread class 4
- Task 2: Create thread objects 4
- Task 3: Suspend the thread operation 4
- Task 4: Execute the solution 4

Exercise 6: Developing Multithreaded Text Extractor Application 5

- Task 1: Task Class Definition 5
- Task 2: Define constructor 5
- Task 3: Random Text Generator 5
- Task 4: Random text extraction 5
- Task 5: Override run ( ) method 6
- Task 6: Create a multithreaded application class 6
- Task 7: Create multiple thread objects 6

TASK 8: EXCUTE MULTITHREADED APPLICATION 6

Multithreading

Learning Outcomes

At the end of this lab, the student should be able to:

1. Define task class using Thread and Runnable.
2. Create multiple threads.
3. Keeping track of thread execution using the thread name.
4. Suspend the execution of the selected thread.

Exercise 1: Developing Simple Multithreading Application using extends Thread

Task 1: Prepare Development

1. Download CurrentTimeTask.java and CurrentTimeMultiThreadsApp.java from ulearn.
2. Put the class in appropriate package in Eclipse.

Task 2: Define Task Class

1. Change the definition of `CurrentTimeTask.java` by extending it to class `Thread`.
2. Save the CurrentTimeTask.java.

Task 3: Define Thread Operation

1. Override the `run( )` method. Refer here for the method definition.
2. The `run( )` method should prints 10 current time using `getCurrentTime( )`.

Task 4: Execute Multithreads Application

1. Execute `CurrentTimeMultiThreadsApp.java` several times.

2. Observe the output, the Java classes, and objects.

Exercise 2: Developing Simple Multithreading Application using extends Thread

Task 1: Prepare Development

1. Download RandomNumberTask.java and RandomNumberMultiThreadsApp.java from ulearn.
2. Put the class in appropriate package in Eclipse.

Task 2: Define Task Class

1. Change the definition of `RandomNumberTask.java` by implementing it to interface `Runnable`.
2. Save the RandomNumberTask.java.

Task 3: Define Thread Operation

1. Override the `run( )` method. Refer here for the method definition.
2. The `run( )` method should prints 10 random numbers using `getRandom( )`.

Task 4: Execute Multithreads Application

1. Execute `RandomNumberMultiThreadsApp.java` for several times.
2. Observe the output, the Java classes, and objects.

Exercise 3: Keeping Track Multiple Threads

Task 1: Name the threads

1. Give two different names for the two thread objects in `CurrentTimeMultiThreadsApp.java`.
2. Use the suitable constructor with parameter and `setName( )` on either one of the thread objects. Refer to the Thread API here.

3. Save CurrentTimeMultiThreadsApp.java.

Task 2: Display the thread's name

1. Amend the implementation of the `run ( )` in `CurrentTimeTask.java` to retrieve the name of a current thread.
2. The `run( )` should should prints 10 statements that consis of the name of the current thread and current time.
3. Save the `CurrentTimeTask.java`.

Task 4: Execute Multithreads Application

1. Execute `CurrentTimeMultiThreadsApp.java` for several times.
2. Observe the output, the Java classes, and objects.

Exercise 4: Keeping Track Multiple Threads

Task 1: Name the threads

1. Give two different names for the two thread objects in `RandomNumberMultiThreadsApp.java`.
2. Use the suitable constructor with parameter and `setName( )` on either one of the thread objects.
3. Save `RandomNumberMultiThreadsApp.java` for several times.

Task 2: Display the thread's name

1. Amend the implementation of the `run ( )` in `RandomNumberTask.java` to retrieve the name of a current thread.
2. The `run( )` should should prints 10 statements that consist of the name of the current thread and random numbers.
3. Save the `RandomNumberTask.java`.

Task 4: Execute Multithreads Application

1. Execute `RandomNumberMultiThreadsApp.java` for several times.
2. Observe the output, the Java classes, and objects.

Exercise 5: Developing Multithread Application on Different Thread Tasks

Task 1: Create new multithread class

1. Create a class named `CurrentTimeRandomNumberApp.java` with a `main( )` method.
2. Save the class.

Task 2: Create thread objects

1. Create 2 objects from `RandomNumberTask.java`.
2. Name the thread objects.
3. Create 1 object from `CurrentTimeTask.java`.
4. Name the thread object.

Task 3: Suspend the thread operation

1. Amend `CurrentTimeTask.java` to suspend a print operation for 4000 milliseconds before it prints the next statement. Use `sleep( )` method.
2. Save CurrentTimeTask.java.
3. Amend `RandomNumberTask.java` to suspend the operation for 5000 milliseconds after it has finished printing all statements.
4. Print another statement to indicate the task is finished after the suspension.
5. Save RandomNumberTask.java.

Task 4: Execute the solution

1. Open CurrentTimeRandomNumberApp.java.
2. Add Java instruction to execute the thread objects created in Task 2 by using `start( )`. Refer to the Thread API here.
3. Execute this class for several times.
4. Observe the output and the Java objects.

Exercise 6: Developing Multithreaded Text Extractor Application

Task 1: Task Class Definition

1. Define a class named `TextExtractorTask`. This class represent a task class.
2. Extend the definition of `TextExtractorTask` using extend `Thread` or implements `Runnable`.
3. This class will have 2 private attributes. The first attributes represent a text. Declare this attribute.
4. The second attribute represents a list of word in a text. Declare this attribute using `List`. Refer to this API for further details of `List`.

Task 2: Define constructor

1. Define a constructor that will construct the attributes defined in Task 1.
2. Assign the first attribute to "The List component presents the user with a scrolling list of text items. The list can be set up so that the user can choose either one item or multiple items."
3. The second attribute will contain each word from the first attribute. Use `ArrayList` to construct the object of second attribute. Refer to this API for further details of `ArrayList`.

Task 3: Random Text Generator

1. Construct a method that will generate text from a list of word.
2. The text will contain words in random order from the second attribute of this class.
3. The method will return the text generated from the random ordered words.

Task 4: Random text extraction

1. Construct a method that will extract a portion of text at random length.
2. The method will return the extracted portion

Task 5: Override `run( )` method

1. Override the `run( )` method.
2. The method will execute the method defined in Task 3 when the name of the thread contains "`Generator`".
3. The method will execute the method defined in Task 4 when the name of the thread contains "`Extractor`". It will display the value from this method.
4. This method will also print the name of the current thread and the value retrieved from the method invoked in number 3 and 4.

Task 6: Create a multithreaded application class

1. Create a class that will represent a multithread application. This class will have a `main( )` method.
2. Name the class appropriately.

Task 7: Create multiple thread objects

1. Create three objects from the class defined in Task 1.
2. Name the objects as "`textExtractor`", "`textGenerator`", "`textAnalyzer`"
3. These objects must be executed.

Task 8: Excute multithreaded application

1. Execute the class defined in Task 6 for several times.
2. Observe the output and the object.