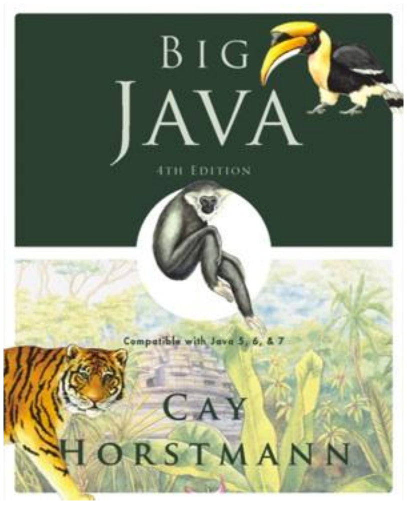

Chapter 20 – Multithreading

Chapter Goals

- To understand how multiple threads can execute in parallel
- To learn how to implement threads

Threads

- **Thread:** a program unit that is executed independently of other parts of the program
- The Java Virtual Machine executes each thread in the program for a short amount of time
- This gives the impression of parallel execution

Running a Thread

- Implement a class that implements the `Runnable` interface:

```
public interface Runnable
{
   void run();
}
```

- Place the code for your task into the `run` method of your class:

```
public class MyRunnable implements Runnable
{
   public void run()
   {
      Task statements
      ...
   }
}
```

Running a Thread

- Create an object of your subclass:

```
Runnable r = new MyRunnable();
```

- Construct a `Thread` object from the `runnable` object:

```
Thread t = new Thread(r);
```

- Call the `start` method to start the thread:

```
t.start();
```

Example

A program to print a time stamp and "Hello World" once a second for ten seconds:

```
Mon Dec 28 23:12:03 PST 2009 Hello, World!
Mon Dec 28 23:12:04 PST 2009 Hello, World!
Mon Dec 28 23:12:05 PST 2009 Hello, World!
Mon Dec 28 23:12:06 PST 2009 Hello, World!
Mon Dec 28 23:12:07 PST 2009 Hello, World!
Mon Dec 28 23:12:08 PST 2009 Hello, World!
Mon Dec 28 23:12:09 PST 2009 Hello, World!
Mon Dec 28 23:12:10 PST 2009 Hello, World!
Mon Dec 28 23:12:11 PST 2009 Hello, World!
Mon Dec 28 23:12:12 PST 2009 Hello, World!
```

`GreetingRunnable` Outline

```
public class GreetingRunnable implements Runnable
{
   private String greeting;

   public GreetingRunnable(String aGreeting)
   {
      greeting = aGreeting;
   }

   public void run()
   {
      Task statements
      ...
   }
}
```

Thread Action for `GreetingRunnable`

- Print a time stamp
- Print the greeting
- Wait a second

GreetingRunnable

- We can get the date and time by constructing a `Date` object:

  ```
  Date now = new Date();
  ```

- To wait a second, use the sleep method of the `Thread` class:

  ```
  sleep(milliseconds)
  ```

- A sleeping thread can generate an `InterruptedException`
  - *Catch the exception*
  - *Terminate the thread*

Running Threads

- `sleep` puts current thread to sleep for given number of milliseconds:

  ```
  Thread.sleep(milliseconds)
  ```

- When a thread is interrupted, most common response is to terminate `run`

Generic `run` method

```
public void run()
{
   try
   {
      Task statements
   }
   catch (InterruptedException exception)
   {
   }
   Clean up, if necessary
}
```

To Start the Thread

- Construct an object of your `runnable` class:

```
Runnable t = new GreetingRunnable("Hello World");
```

- Then construct a thread and call the `start` method:

```
Thread t = new Thread(r);
t.start();
```

ch20/greeting/GreetingThreadRunner.java (cont.)

Program Run:

```
Mon Dec 28 12:04:46 PST 2009 Hello, World!
Mon Dec 28 12:04:46 PST 2009 Goodbye, World!
Mon Dec 28 12:04:47 PST 2009 Hello, World!
Mon Dec 28 12:04:47 PST 2009 Goodbye, World!
Mon Dec 28 12:04:48 PST 2009 Hello, World!
Mon Dec 28 12:04:48 PST 2009 Goodbye, World!
Mon Dec 28 12:04:49 PST 2009 Hello, World!
Mon Dec 28 12:04:49 PST 2009 Goodbye, World!
Mon Dec 28 12:04:50 PST 2009 Hello, World!
Mon Dec 28 12:04:50 PST 2009 Goodbye, World!
Mon Dec 28 12:04:51 PST 2009 Hello, World!
Mon Dec 28 12:04:51 PST 2009 Goodbye, World!
Mon Dec 28 12:04:52 PST 2009 Goodbye, World!
Mon Dec 28 12:04:52 PST 2009 Hello, World!
Mon Dec 28 12:04:53 PST 2009 Hello, World!
Mon Dec 28 12:04:53 PST 2009 Goodbye, World!
Mon Dec 28 12:04:54 PST 2009 Hello, World!
Mon Dec 28 12:04:54 PST 2009 Goodbye, World!
Mon Dec 28 12:04:55 PST 2009 Hello, World!
Mon Dec 28 12:04:55 PST 2009 Goodbye, World!
```

Thread Scheduler

- **Thread scheduler:** runs each thread for a short amount of time (a **time slice**)
- Then the scheduler activates another thread
- There will always be slight variations in running times - especially when calling operating system services (e.g. input and output)
- There is no guarantee about the order in which threads are executed

Self Check 20.1

What happens if you change the call to the `sleep` method in the `run` method to `Thread.sleep(1)`?

Answer: The messages are printed about one millisecond apart.

Self Check 20.2

What would be the result of the program if the `main` method called

```
r1.run();
r2.run();
```

instead of starting threads?

Answer: The first call to `run` would print ten “Hello” messages, and then the second call to `run` would print ten “Goodbye” messages

Terminating Threads

- A thread terminates when its `run` method terminates
- Do not terminate a thread using the deprecated `stop` method
- Instead, notify a thread that it should terminate:

```
t.interrupt();
```

- `interrupt` does not cause the thread to terminate – it sets a boolean variable in the thread data structure

Terminating Threads

- The `run` method should check occasionally whether it has been interrupted
  - *Use the `interrupted` method*
  - *An interrupted thread should release resources, clean up, and exit:*

```
public void run()
{
   for (int i = 1;
      i <= REPETITIONS && !Thread.interrupted();
      i++)
   {
      Do work
   }
   Clean up
}
```

Terminating Threads

- The `sleep` method throws an `InterruptedException` when a sleeping thread is interrupted
  - *Catch the exception*
  - *Terminate the thread :*

```
public void run()
{
   try
   {
      for (int i = 1; i <= REPETITIONS; i++)
      {
          Do work
          Sleep
      }
   }
   catch (InterruptedException exception)
   {
        Clean up
   }
}
```

Terminating Threads

- Java does not force a thread to terminate when it is interrupted
- It is entirely up to the thread what it does when it is interrupted
- Interrupting is a general mechanism for getting the thread’s attention

Self Check 20.3

Suppose a web browser uses multiple threads to load the images on a web page. Why should these threads be terminated when the user hits the “Back” button?

> **Answer:** If the user hits the “Back” button, the current web page is no longer displayed, and it makes no sense to expend network resources for fetching additional image data.