

Bài 18

Threading

Module: ADVANCED PROGRAMMING WITH JAVA

Mục tiêu

- Trình bày được cơ chế hoạt động của Thread
- Trình bày được cơ chế multi-threading
- Triển khai được cơ chế multi-threading
- Triển khai được cơ chế đồng bộ trong các ứng dụng multi-threading

Thread

Cơ chế multi-threading

Tạo Thread

Vòng đời Thread

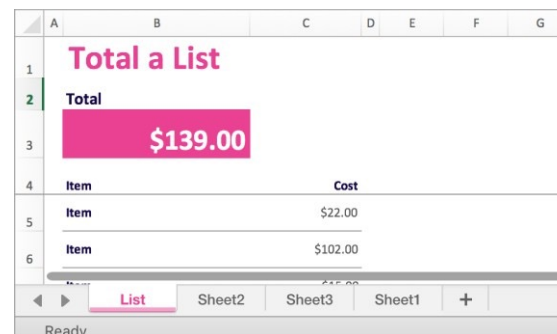
Đồng bộ hoá

Khái niệm Multi-Tasking và Multi-Threading

- Multi-Tasking là khả năng chạy đồng thời nhiều chương trình cùng một lúc trên hệ điều hành
 - Ví dụ như có thể bật nhiều chương trình PowerPoint, Chrome, IntelliJ... một lúc



- Multi-Threading là khả năng thực hiện đồng thời nhiều tiểu trình trong một chương trình.
 - Ví dụ như trong Excel có thể đồng thời có nhiều sheet được tạo và thao tác 1 lúc.

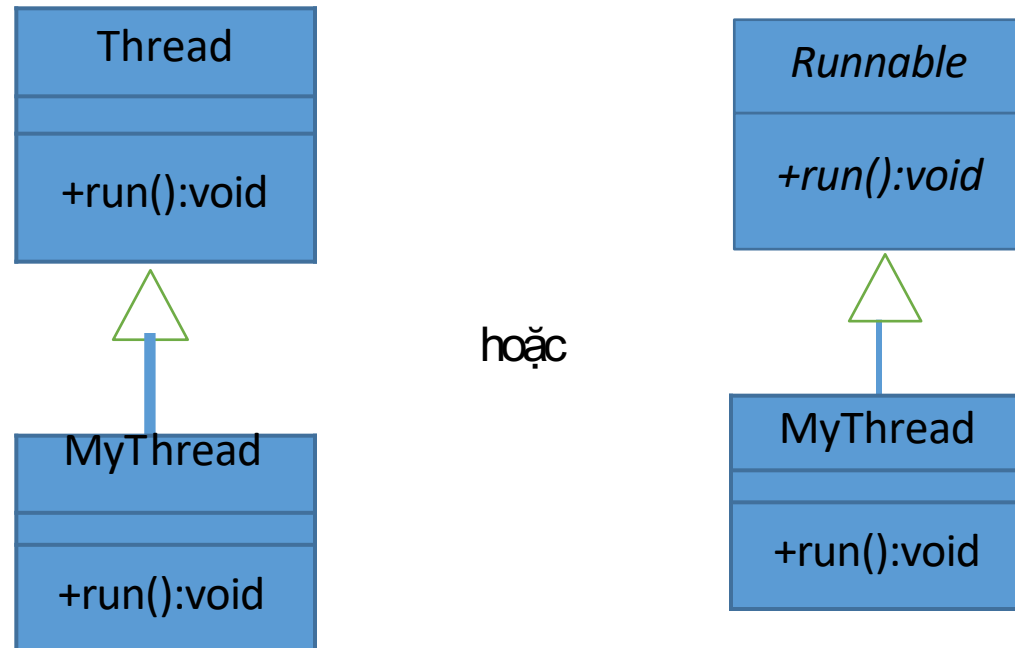


Thread là gì?

- Thread là đơn vị nhỏ nhất của mã thực thi mà đoạn mã đó thực hiện một nhiệm vụ cụ thể.
- Một ứng dụng có thể được chia nhỏ thành nhiều nhiệm vụ và mỗi nhiệm vụ có thể được giao cho một thread.
- Nhiều thread cùng thực hiện đồng thời được gọi là đa luồng (Multi-Thread)
- Các quá trình đang chạy dường như là đồng thời, nhưng thực ra nó không phải là như vậy.

Tạo Thread

- Hệ thống đa luồng trong Java được xây dựng trên lớp Thread và interface Runnable thuộc gói java.lang
- Có 2 cách để tạo Thread mới
- Cách 1: Kế thừa từ lớp Thread có sẵn
- Cách 2: Thực thi interface Runnable có sẵn



Tạo Thread qua kế thừa từ lớp Thread

- Bước 1: Tạo lớp MyThread kế thừa từ lớp Thread
- Bước 2: Override phương thức run()

```
// Extending Thread class
class MyThread extends Thread {
    public void run()    { // Overriding the Run()
        // implementation the logic
    }
}
```

- Bước 3: Tạo đối tượng thread và gọi phương thức start()

```
public static void main(String args[]) {
    //Creating thread object
    MyThread t = new MyThread();
    t.start(); //Starting a thread
}
```

Tạo Thread qua thực thi interface Runnable

- Bước 1: Tạo lớp MyRunnable thực thi interface Runnable
- Bước 2: Implement phương thức run()

```
// Declaring a class that implements Runnable interface  
class MyRunnable implements Runnable {  
    public void run()    {// Overriding the Run()  
        // implementation the logic  
    }  
}
```

- Bước 3: Tạo đối tượng thread và gọi phương thức start()

```
public static void main(String args[]) {  
    Runnable r = new MyRunnable();  
    Thread thObj=new Thread(r);  
    thObj.start(); //Starting a thread  
}
```


Ví dụ: Chương trình tạo 3 task ứng với 3 thread



- Task1: In ký tự a 100 lần
- Task2: In ký tự b 100 lần
- Task3: In số nguyên từ 1 đến 100

```
// The task class for printing numbers from 1 to n for a given n
class PrintNum implements Runnable {
    private int lastNum;

    /** Construct a task for printing 1, 2, ..., n */
    public PrintNum(int n) {
        lastNum = n;
    }

    @Override /** Tell the thread how to run */
    public void run() {
        for (int i = 1; i <= lastNum; i++) {
            System.out.print(" " + i);
        }
    }
}
```

```
// The task for printing a character a specified number of times
class PrintChar implements Runnable {
    private char charToPrint; // The character to print
    private int times; // The number of times to repeat

    /** Construct a task with a specified character and number of
     * times to print the character
     */
    public PrintChar(char c, int t) {
        charToPrint = c;
        times = t;
    }

    @Override /** Override the run() method to tell the system
     * what task to perform
     */
    public void run() {
        for (int i = 0; i < times; i++) {
            System.out.print(charToPrint);
        }
    }
}
```

Ví dụ: Chương trình tạo 3 task ứng với 3 thread

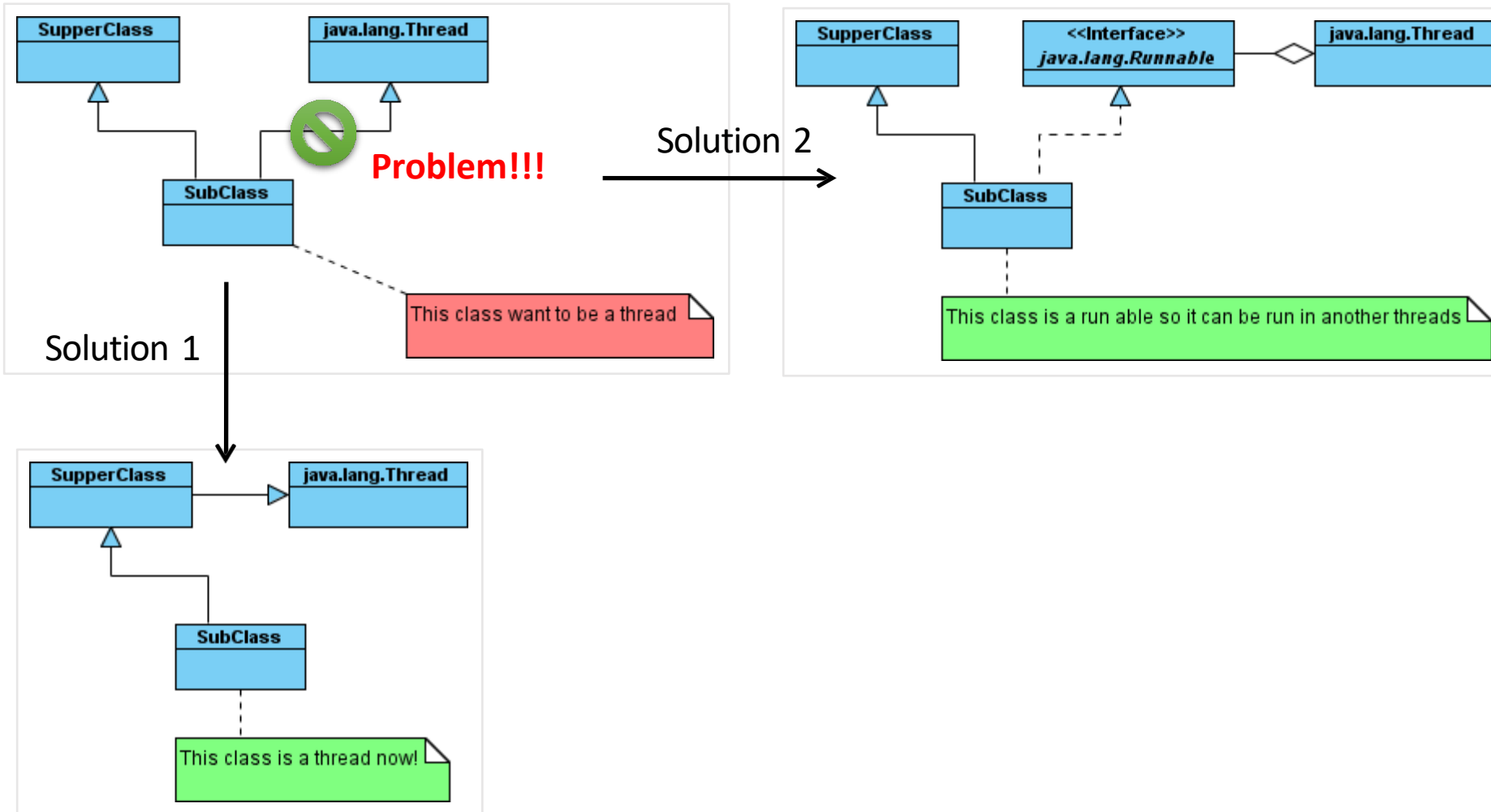


- Tạo 3 task và thực thi

```
public class TaskThreadDemo {  
    public static void main(String[] args) {  
        // Create tasks  
        Runnable printA = new PrintChar('a', 100);  
        Runnable printB = new PrintChar('b', 100);  
        Runnable print100 = new PrintNum(100);  
  
        // Create threads  
        Thread thread1 = new Thread(printA);  
        Thread thread2 = new Thread(printB);  
        Thread thread3 = new Thread(print100);  
  
        // Start threads  
        thread1.start();  
        thread2.start();  
        thread3.start();  
    }  
}
```

Lưu ý: Phương thức run() gọi tự động khi phương thức start() được thực thi.

Sự khác nhau giữa hai cách tạo Thread



Tạo Thread với kỹ thuật Anonymous class

- Anonymous class với Thread

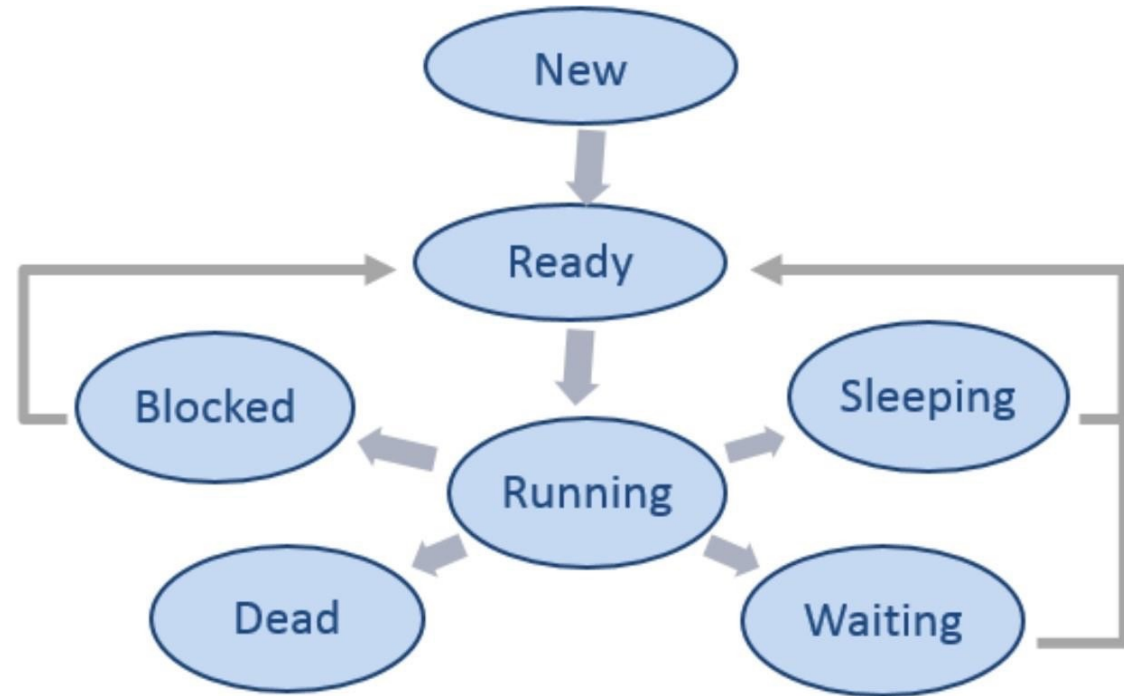
```
new Thread(){  
    public void run(){}  
}.start();
```

- Anonymous class với Runnable

```
new Thread(new Runnable(){  
    public void run(){}  
}).start();
```

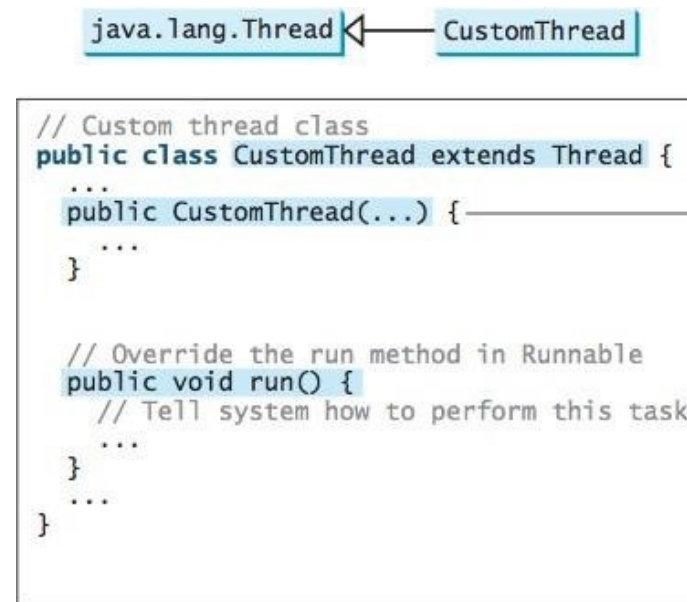
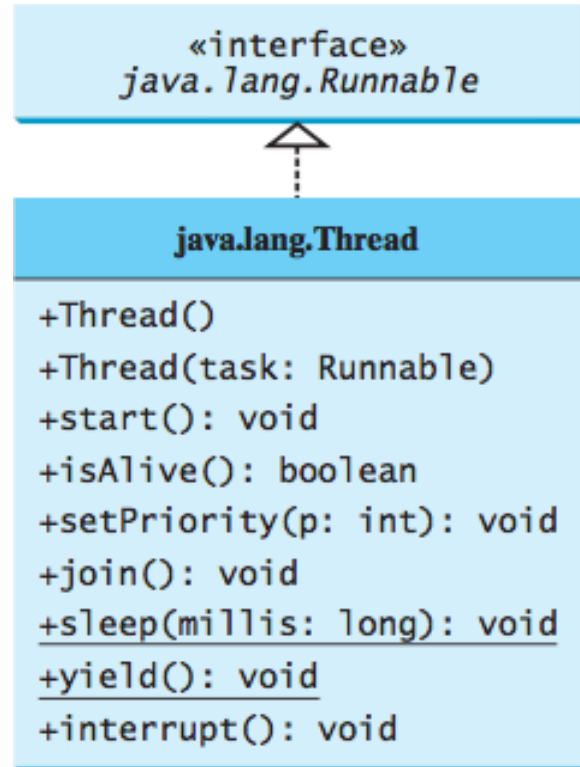
Vòng đời của một Thread

- new
- runnable (ready to run)
- running
- sleeping
- waiting
- blocked
- dead



Lớp Thread trong java

- Thread thuộc gói java.lang trong Java.



```
classDiagram
    class Thread {
        java.lang.Thread
    }
    class CustomThread {
    }
    Thread <|-- CustomThread
```

java.lang.Thread ← **CustomThread**

```
// Custom thread class
public class CustomThread extends Thread {
    ...
    public CustomThread(...) {
        ...
    }
    ...
    // Override the run method in Runnable
    public void run() {
        // Tell system how to perform this task
        ...
    }
    ...
}
```

(a)

```
// Client class
public class Client {
    ...
    public void someMethod() {
        ...
        // Create a thread
        CustomThread thread1 = new CustomThread(...);

        // Start a thread
        thread1.start();
        ...

        // Create another thread
        CustomThread thread2 = new CustomThread(...);

        // Start a thread
        thread2.start();
    }
    ...
}
```

(b)

Đồng bộ hoá (Synchronization)

- Nếu nhiều thread đang hoạt động đồng thời mà sử dụng chung một tài nguyên nào đó thì sẽ xảy ra xung đột.
- Đồng bộ hoá chính là việc sắp xếp thứ tự các thread khi truy xuất vào cùng đối tượng sao cho không có sự xung đột dữ liệu.
- Để đảm bảo rằng một nguồn tài nguyên chia sẻ được sử dụng bởi một thread tại một thời điểm, chúng ta sử dụng đồng bộ hoá.

Đồng bộ hoá (Synchronization)

- Một "monitor" là một công cụ giám sát hỗ trợ cho việc đồng bộ hoá các luồng.
- Tại một thời điểm chỉ có một thread được vào "monitor"
- Khi một thread vào được "monitor" thì tất cả các thread khác sẽ phải đợi đến khi thread này ra khỏi "monitor"
- Để đưa một thread vào monitor, chúng ta phải gọi một phương thức có sử dụng từ khoá synchronized.
- Sau khi thread đang chiếm giữ monitor này kết thúc công việc và thoát khỏi monitor thì luồng tiếp theo mới có thể "vào được" monitor.

Đồng bộ hoá (Synchronization)

- Khi nhiều thread cùng gọi một phương thức được khai báo với synchronized thì cái gọi sau sẽ phải đợi

```
public class MyRunnable implements Runnable{  
    @Override  
    public void run() {...}  
}
```

→ t1 và t2 chạy đồng thời

Cả 2 thread t1 và t2 dùng chung run

```
MyRunnable run = new MyRunnable();  
Thread t1 = new Thread(run);  
Thread t2 = new Thread(run);  
t1.start();  
t2.start();
```

```
public class MyRunnable implements Runnable{  
    @Override  
    public synchronized void run() {...}  
}
```

→ t1 chạy xong mới đến t2

Đồng bộ hoá Block

- Đồng bộ hoá một đoạn code trong một phương thức của một đối tượng bằng cách sử dụng từ khoá synchronized.
- Với việc đồng bộ hoá block, chúng ta có thể khoá chính xác đoạn code mình cần.
- Đồng bộ hoá phương thức có thể được viết lại bằng đồng bộ hoá block như sau

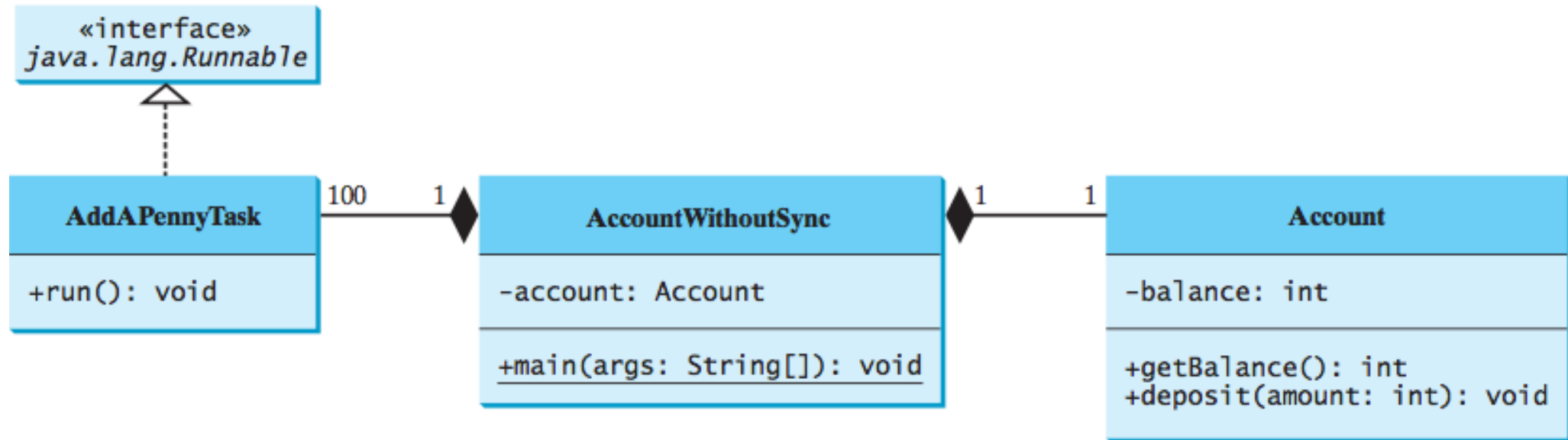
```
public synchronized void run(){  
    .....  
}
```



```
public void run(){  
    synchronized(this){  
        ...  
    }  
}
```

Ví dụ

- Tạo ứng dụng thực hiện rút tiền



Ví dụ

```
import java.util.concurrent.*;

public class AccountWithoutSync {
    private static Account account = new Account();

    public static void main(String[] args) {
        ExecutorService executor = Executors.newCachedThreadPool();

        // Create and launch 100 threads
        for (int i = 0; i < 100; i++) {
            executor.execute(new AddAPennyTask());
        }

        executor.shutdown();

        // Wait until all tasks are finished
        while (!executor.isTerminated()) {
        }

        System.out.println("What is balance? " + account.getBalance());
    }

    // A thread for adding a penny to the account
    private static class AddAPennyTask implements Runnable {
        public void run() {
            account.deposit(1);
        }
    }
}
```

```
// An inner class for account
private static class Account {
    private int balance = 0;

    public int getBalance() {
        return balance;
    }

    public void deposit(int amount) {
        int newBalance = balance + amount;

        // This delay is deliberately added to magnify the
        // data-corruption problem and make it easy to see.
        try {
            Thread.sleep(5);
        }
        catch (InterruptedException ex) {
        }

        balance = newBalance;
    }
}
```

- Multi-Threading là khả năng thực hiện đồng thời nhiều tiểu trình trong một chương trình.
- Thread là đơn vị nhỏ nhất của mã thực thi mà đoạn mã đó thực hiện một nhiệm vụ cụ thể.
- Có 2 cách để tạo Thread mới
- Cách 1: Kế thừa từ lớp Thread có sẵn
- Cách 2: Thực thi interface Runnable có sẵn
- Đồng bộ hoá chính là việc sắp xếp thứ tự các thread khi truy xuất vào cùng đối tượng sao cho không có sự xung đột dữ liệu.
- Một "monitor" là một công cụ giám sát hỗ trợ cho việc đồng bộ hoá các luồng.

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: String