Chp12 线程

Key Point

- 线程的概念
- 线程的创建
- 线程的状态转换
- 线程间数据共享
- 线程的同步

练习

1. 一个单CPU 的机器，如何同时执行多个线程？请简述其原理

2. （线程的创建）有以下代码

```
public class Example implements Runnable {
    public void run() {
        while(true) {
        }
    }
    public static void main(String args[]) {
        Example ex1 = new Example();
        Example ex2 = new Example();
        Example ex3 = new Example();
        ex1.run();
        ex2.run();
        ex3.run();
    }
}
```

选择正确答案：

A. 代码编译失败，因为ex2.run()无法获得执行
B. 代码编译成功，存在3 个可运行的线程
C. 代码编译成功，存在1 个可运行的线程

3. （线程的创建）有以下代码

```
class Example implements Runnable {
    public static void main(String args[]) {
        Thread t = new Thread(new Example());
        t.start();
    }
    public void run(int limit) {
        for (int x = 0; x<limit; x++) {
            System.out.println(x);
        }
```

```
    }
}
```
选择正确答案：

A. 打印输出，从0 至limit

B. 无内容输出，因为没有明确调用run()方法。

C. 代码编译失败，因为没有正确实现Runnable 接口

D. 代码编译失败，如果声明类为抽象类，可使代码编译成功。

E. 代码编译失败，如果去掉implements Runnable，可使代码编译成功。

4. （sleep 方法）有如下代码

```
class Example {
    public static void main(String args[]) {
        Thread.sleep(3000);
        System.out.println(“sleep”);
    }
}
```
选择正确答案：

A. 编译出错

B. 运行时异常

C. 正常编译运行，输出sleep

D. 正常编译运行，但没有内容输出

5. （线程的创建）有如下代码

```
class Example extends Thread {
    public void run() {
        System.out.println(“Running”);
        System.out.println(“Done”);
    }
    private void xxx() {
        System.out.println(“In xxx”);
    }
    public static void main(String args[]){
        Example ttt = new Example();
        ttt.start();
        ttt.xxx();
    }
}
```
以上代码是否会打印“In xxx”？

6. （线程的创建）创建两个线程，要求如下：

1） 一个线程输出100 个1~26，另一个线程输出100 个A~Z

2） 一个线程使用继承Thread 类的写法，另一个线程使用实现Runnable 接口的写法。

7. （线程状态）有如下代码

```
class MyThread extends Thread{
```

```
    public void run(){
        for(int i = 0; i<100; i++){
            System.out.println(“$$$”);
        }
    }
}
public class TestMyThread{
    public static void main(String args[]){
        Thread t = new MyThread();
        t.start();
        t.start();
        t.start();
    }
}
```

选择正确答案：

A. 编译出错

B. 编译正常，运行时有错

C. 编译运行都无错误，产生1 个线程

D. 编译运行都无错误，产生3 个线程

8. （线程的同步）有如下代码

```
class MyThread1 extends Thread{
    Object lock;
    public MyThread1(Object lock){
        this.lock = lock;
    }
    public void run(){
        synchronized(lock){ //1
            for(int i = 0; i<=10; i++){
                try{
                    Thread.sleep( (int)(Math.random()*1000) );
                }catch(Exception e){}
                System.out.println(“$$$”);
            }
        }
    }
}
class MyThread2 extends Thread{
    Object lock;
    public MyThread2(Object lock){
        this.lock = lock;
    }
    public void run(){
        synchronized(lock){ //2
            for(int i = 0; i<=10; i++){
```

```
                try{
                    Thread.sleep((int)(Math.random()*1000) );
                }catch(Exception e){}
                System.out.println(“###”);
            }
        }
    }
}
public class TestMyThread{
    public static void main(String args[]){
        Object lock = new Object();
        Thread t1 = new MyThread1(lock);
        Thread t2 = new MyThread2(lock);
        t1.start();
        t2.start();
    }
}
```

问：在//1 和//2 处加上的synchronized 起什么作用？如果不加synchronized，运行程序有什么不同的地方？

9. *（线程同步）有如下代码

```
class MyThread extends Thread{
    private String data;
    public void run(){
        synchronized(data){
            for(int i = 0; i<10; i++){
                try{
                    Thread.sleep((int)(Math.random()*1000) );
                }catch(Exception e){}
                System.out.println(data);
            }
        }
    }
}
public class TestMyThread {
    public static void main(String args[]){
        Thread t1 = new MyThread(“hello”);
        Thread t2 = new MyThread(“world”);
        t1.start();
        t2.start();
    }
}
```

问：上述代码输出的结果是什么？

A. 先输出100 个hello，然后是100 个world

B. 先输出100 个world，然后是100 个hello

C. 线程不同步，因此交替输出hello 和world

10. *（线程同步）有下面代码

```
1) class MyThread extends Thread{
2)   private String data;
3)   public MyThread(String data){
4)       this.data = data;
5)   }
6)   public void run(){
7)       for(int i = 0; i<100; i++){
8)           System.out.println(data);
9)       }
10)  }
11) }
12) public class TestMyThread{
13)    public static void main(String args[]){
14)        Thread t1 = new MyThread(“aaa”);
15)        Thread t2 = new MyThread(“bbb”);
16)        t1.start();
17)        t2.start();
18)    }
19) }
```

现希望能够同步的输出aaa 和bbb，即一次输出100 个aaa 或bbb，输出这两个字符串时没有交互。

为了达到上述目的，要对原代码进行修改。以下哪些修改方式能够得到想要的结果？

A. 把第6 行改为public synchronized void run()

B. 把run 方法中所有的内容都放在synchronized(data)代码块中

C. 把run 方法中所有的内容都放在synchronized(System.out)代码块中

11. *（线程的同步）有如下代码

```
class MyThread extends Thread{
    private int data;
    public void run(){
        data += 2;
        System.out.print(data + “ ”);
    }
}
public class TestMyThread{
    public static void main(String args[]){
        Thread t1 = new MyThread();
        Thread t2 = new MyThread();
        Thread t3 = new MyThread();
        t1.start();
        t2.start();
```

```
        t3.start();
        System.out.println();
    }
}
```
写出上述程序运行的结果。

12. *（线程同步）有如下代码
```
class MyValue{
    private int data;
    public void m(){
        int result = 0;
        result += 2;
        data += 2;
        System.out.println(result + " " + data);
    }
}
class MyThread extends Thread{
    private MyValue mv;
    public MyThread(MyValue mv){
        this.mv = mv;
    }
    public void run(){
        synchronized(mv){
            mv.m();
        }
    }
}
public class TestMyThread{
    public static void main(String args[]){
        MyValue mv = new MyValue();
        Thread t1 = new MyThread(mv);
        Thread t2 = new MyThread(mv);
        Thread t3 = new MyThread(mv);
        t1.start();
        t2.start();
        t3.start();
    }
}
```
问：

1） 写出该程序输出的结果

2） 如果把run 方法中的synchronized 代码块去掉，而是直接调用
```
public void run(){
    mv.m();
}
```

则会产生同步问题。如果不允许改动MyThread 类，应该如何修改，以保证代码同步？

13. *（线程综合）代码改错

```
class MyThread1 implements Runnable{
    public void run() {
        for(int i = 0; i<100; i++){
            this.sleep((int)(Math.random()*1000));
            System.out.println(“hello”);
        }
    }
}
class MyThread2 extends Thread{
    public void run() throws Exception {
        for(int i = 0; i<100; i++){
            this.sleep((int)(Math.random()*1000));
            System.out.println(“world”);
        }
    }
}
public class TestMyThread{
    public static void main(String args[]){
        Runnable t1 = new MyThread1();
        Thread t2 = new MyThread2();
        t1.start();
        t2.start();
    }
}
```

14. **（线程的同步）有如下代码

```
class MyThread extends Thread{
    private int data;
    public int getData() {
        return data;
    }
    public void setData(int data) {
        this.data = data;
    }
    public synchronized void run(){
        for(int i = 0; i<100; i++){
            System.out.println(data);
            try {
                Thread.sleep((int)(Math.random()*1000));
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
            }
        }
    }
}
public class TestThread {
    public static void main(String[] args) throws Exception{
        MyThread t1 = new MyThread();
        t1.setData(100);
        t1.start();
        Thread.sleep(5000);
        t1.setData(200);
    }
}
```

问：在t1 线程启动以后，主方法中调用setData 方法，能否修改t1 线程中的data值？如果可以，应该如何修改代码，使得在t1 运行过程中，其data 属性不会被修改？如果不可以，画出线程运行的状态示意图。

15. **（多线程）完成下列程序要求

有个Student 类，代码如下：

```
class Student{
    String name;
    int age;
    //构造方法和get/set 方法请自行补充完成
    …
    //学生问老师问题
    public void ask(Teacher t){
        t.answer(this);//调用老师的answer 方法
    }
    public void study(){
        System.out.println(name + “ study”);
    }
    public void doHomework(){
        System.out.println(name + “ do homework”);
    }
}
```

定义Teacher 接口

```
interface Teacher{
    void answer(Student stu);
}
```

给出一个Teacher 接口的实现类。该实现类实现answer 方法的时候，要求每次学生调用老师的answer 方法时，都创建一个新线程，该线程调用学生的学习方法和做作业方法。

16. **（线程的同步、数据共享）有如下代码

```
class Computation implements Runnable {
    private int result;
    public Computation() {
    }
    public void run() {
        countprint(this.result);
    }
    public synchronized void countprint(int result) {
        result = result + 2;
        System.out.println(result);
    }
}
public class TestComputation{
    public static void main(String args[]){
        Runnable target = new Computation();
        new Thread(target).start();
        new Thread(target).start();
        new Thread(target).start();
        new Thread(target).start();
    }
}
```

写出该程序的运行结果。