COMPSCI 230 Threading Week7

“In computer science, a thread of execution is the smallest sequence of programmed instructions that can be managed independently by an operating system scheduler. The scheduler itself is a light-weight process. The implementation of threads and processes differs from one operating system to another, but in most cases, a thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory, while different processes do not share these resources”.

Figure 1 shows using quote core processors to run multiple threads

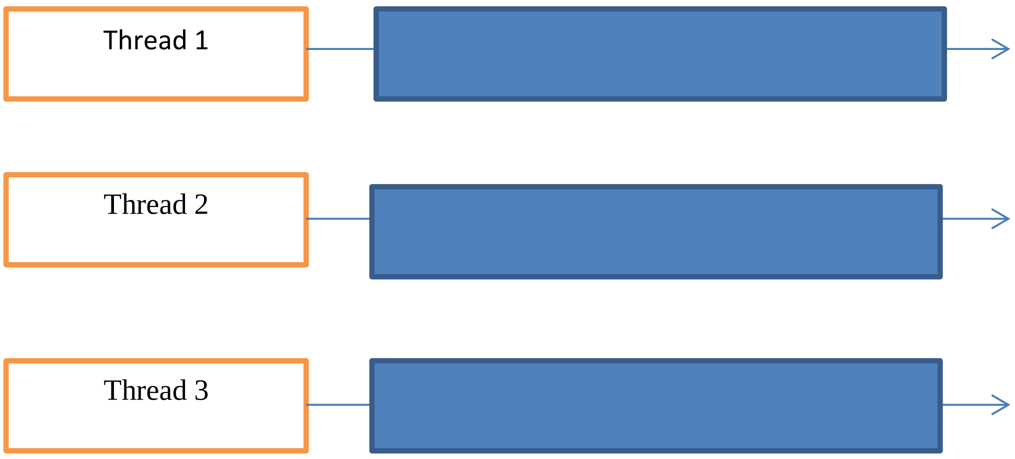
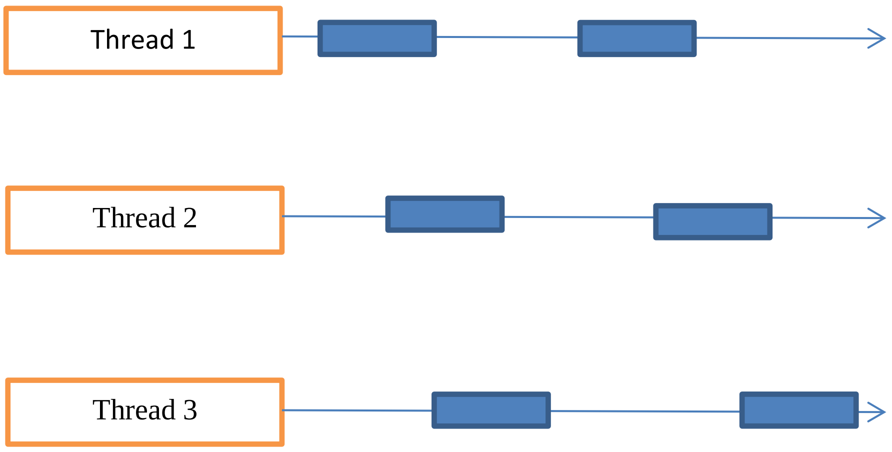


Figure 2 shows using a single processor to run multiple threads



Modify the testthread.java to create three threads, thread 1 will print the character 'a' 100 times, thread 2 will print the character 'b' 100 times and thread 3 will print the number from 1 to 100.
Q1: Please add the codes to create three threads and start threads

```
public class TestThread
{
  /**Main method*/
  public static void main(String[] args)
  {
    // Create threads




    // Start threads




  }
}
```

Q2: Please complete the blank part for the following line

```
class PrintNum extends
```

```
class PrintChar extends
```

Q3: Please override the run() method in both PrintChar and PrintNum classes

```
class PrintNum
{
  public void run()
  {




  }
}
```

```
class PrintChar
{
  public void run()
  {






  }
}
```

Q2: InTestRunnable.java, use the Runnable interface to create three threads, thread 1 will print the character ‘a’ 100 times, thread 2 will print the character ‘b’ 100 times and thread 3 will print the number from 1 to 100.

Q3: In TestRunnableSleep.java, modify the PrintNum class to allow the thread to sleep 1s, if the next print number is bigger than 50.

```
public void run()
  {
    for (int i=1; i <= lastNum; i++)
        //sleep here
      System.out.print(" " + i);

  }
```

Q4: When executing the programs from Q1,Q2, and Q3, the results are unpredictable. Why does this occur?

Q5: What is a major benefit from using Runnable instead of Thread?

Q6: What are the two types of threads and what is the difference?

Q7: Why shouldn't you execute time-consuming tasks in the Event-dispatch thread?

In MyForm.java in package Q4 we have a program that uses SwingWorker to complete background tasks. We're going to add our own background task that selects random letters from the alphabet and creates a string of length 10.

Q8: Create a new class called SwingAlphabet that extends SwingWorker. It should have a private instance variable of type String that is the alphabet. You must override the doInBackground method, the done method, and the process method. It should return a String at the end and print a Character as it's selected.

Q9: Implement code in doInBackground() that: selects a random characters from variable alpha, publishes it so we can display it in the GUI, concatenates it to the end of a String, and finally sleeps for 1 second. Repeat this procedure 10 times then return the new random string.

Q10: Implement code in done() that appends the completed string to the TraceBox.

Q11: Implement code in process() that appends the currently selected letter to Tracebox.

Q12: Finally, add a button to start a new SwingALphabet and change the size of the JFrame to 800,520.

Q13: Look at the AccountWithoutSync.java in package Q5, the customer wants to create 100 threads, and using each thread to add one penny. After 100 threads, the balance should equal to 100.

```
package Q1;

public class AccountWithoutSync {

    private Account account = new Account();
    private Thread[] thread = new Thread[100];

    public static void main(String[] args) {
       AccountWithoutSync test = new AccountWithoutSync();
        System.out.println("What is balance ? " +  test.account.getBalance());
    }

   public AccountWithoutSync(){
           ThreadGroup g = new ThreadGroup("group");
           boolean done = false;
           for(int i =0 ; i< 100; i++)
           {
                  thread[i] = new Thread(g,new AddAPennyThread(), "t");
                  thread[i].start();
           }

           while (!done)
                  if(g.activeCount() == 0)
                         done = true;
   }
    // An inner class of task for adding a penny to the account
    class AddAPennyThread extends Thread {

        public void run() {
            account.deposit(1);
        }
    }
     class Account {

        private int balance = 0;

        public int getBalance() {
            return balance;
        }

        public void deposit(int amount) {

            int newBalance = balance + amount;

            try {
                Thread.sleep(5);
            } catch (InterruptedException ex) {
                // do nothing
            }

            balance = newBalance;
        }
    }
}
```

Q13-a: When you run `AccountWithoutSync.java`, what's the balance?

Q13-b: what's the problem?

Q13-c: How to fix this problem?