THỰC HÀNH BUỔI 1+2 (Thread)

Bài 1: Tạo project có tên là **Bai1**

- Tạo thread có tên CreateThread kế thừa từ java.lang.Thread

```
public class CreateThread extends Thread{
    private String prefix="";
    public CreateThread(String p)
    {
        prefix=p;
    }
    public void run()
    {
        for(int i=0;i<100;i++)
            System.out.println(prefix+i);
    }
    public static void main(String[] args) {
        CreateThread t1=new CreateThread("Luồng thứ 1:");
        CreateThread t2=new CreateThread("Luồng thứ 2:");
        CreateThread t3=new CreateThread("Luồng thứ 3:");
        t1.start();
        t2.start();
        t3.start();
    }
}
```

- Tạo thread có tên là Thraed_runnable thực thi từ Interface Runnable

```
public class Thread_runnable implements Runnable{
    private String prefix="";
    public Thread_runnable(String p)
    {
        prefix=p;
    }
    public void run()
    {
        for(int i=0;i<100;i++)
            System.out.println(prefix+i);
    }
    public static void main(String[] args) {
        Thread_runnable r1=new Thread_runnable("Luồng thứ 1:");
        Thread_runnable r2=new Thread_runnable("Luồng thứ 2:");
        Thread_runnable r3=new Thread_runnable("Luồng thứ 3:");
        Thread t1=new Thread(r1);
        Thread t2=new Thread(r2);
        Thread t3=new Thread(r3);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Bài 2: Tạo project có **Bai2**

- Cho người dùng nhập vào một số nguyên trong khoảng từ ***1*** đến ***100***. Có 3 thread chạy song song cùng tạo ra con số random trong khoảng ***1*** đến ***100*** . Nếu thread nào random ra một số trùng với số mà người chơi vừa nhập → thread đó dừng lại và in ra số lần *"đoán"* . Kết quả tìm ra thread nào đoán nhanh nhất.

```
import java.util.Scanner;
public class Thread_DoanSo extends Thread {
        private int guessNumber = 0;
        private int count = 0;

        public Thread_DoanSo(int guessNumber) {
            this.guessNumber = guessNumber;
        }

        public void run() {
            int randomNumber = 0;
            do {
                randomNumber = (int) (Math.random() * 100 + 1);
                count++;
                System.out.println(getName() + " đoán số " + randomNumber);
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            } while (randomNumber != guessNumber);
            System.out.println(getName() + " đã đoán ra số " + guessNumber + " trong " +
count + " lần đếm");
        }

        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Nhập một số nguyên để các thread đoán: ");
            int number = scanner.nextInt();
            Thread_DoanSo thread1 = new Thread_DoanSo(number);
            Thread_DoanSo thread2 = new Thread_DoanSo(number);
            Thread_DoanSo thread3 = new Thread_DoanSo(number);
            thread1.setName("Thread 1");
            thread2.setName("Thread 2");
            thread3.setName("Thread 2");
            thread1.start();
            thread2.start();
            thread3.start();
        }
}
```

Bài 3: Tạo project có **Bai3.** Viết chương trình tạo ra 3 thread

- *Thread1*: cứ mỗi 2 giây sẽ sinh ra 1 số nguyên ngẫu nhiên trong khoảng từ 1 đến 20. Chạy vô tận.
- *Thread2:* cứ mỗi 1 giây sẽ lấy số ngẫu nhiên mà Thread1 sinh ra rồi tính bình phương của nó và hiện thị ra màn hình. Chạy vô tận.
- *Thread Main*: tạo và khởi động 2 Thread này.

Bài 4: Tạo project có **Bai4.** Tạo 2 thread song song.

- Tạo lớp Thread1 kế thừa Thread gồm các thuộc tính ArrayList<Integer> list1. Thread này thực hiện sau 1s sinh ngẫu nhiên các số tự nhiêu chạy từ 0-100 -> lưu kết quả vào mảng list1
- Tạo lớp Thread2 kế thừa Thread gồm các thuộc tính ArrayList<Character> list2. Thread này thực hiện sau 2s thì sinh ngẫu nhiên các ký tự từ a-z và lưu vào mảng list2
- Trong phương thức main của lớp Test tạo ra 2 thread t1 và t2 lần lượt từ Thread1 và Thread2
  - Khi t1 đã sinh 10 lần -> thực hiện dừng t1
  - Khi t2 đã sinh đc 10 ký tự thì stop t2
  - Sau khi 2 thread 1 và 2 kết thúc. Thực hiện in kết quả của các thread trên thread main

Hướng dẫn:

```
public class Test_thread {
    public static void main(String[] args) throws InterruptedException {
        System.out.println("\n Main Bắt đầu ");
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();
         t1.start();
         t2.start();
         t1.join();
         t2.join();
         System.out.println("Hien thi ket quả sinh ngau nhien 2 so 1 và 2
         System.out.println("\nT1: ");
         for(int i = 0; i < t1.list.size(); i++) {
             System.out.print(" " + t1.list.get(i));
         }
         System.out.println("\nT2: ");
         for(int i = 0; i < t2.list.size(); i++) {
             System.out.print(" " + t2.list.get(i));
         }
         System.out.println("\n Main kết thúc ");
    }
}
```

```java
import java.util.*;
public class Thread1 extends Thread{
    List<Integer> list = new ArrayList<>();
    public void run() {
        Random random = new Random();
        for(int i = 0; i < 10; i++) {
            int rad = random.nextInt(10);
            list.add(rad);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

```java
import java.util.*;
public class Thread2 extends Thread{
    List<Character> list = new ArrayList<>();
    public void run() {
        int min = (int) 'a';
        int max = (int) 'z';
        int limit = max - min;
        Random random = new Random();
        for(int i = 0; i < 10; i++) {
            int rad = random.nextInt(limit) + min;
            char c = (char) rad;
            list.add( c);
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

Bài 5: Tạo project có **Bai5.** Viết chương trình mô phỏng máy rút tiền ATM

- Mỗi khách hàng có tài khoản với số tiền balance
- Cho phép nhiều thao tác rút hoặc nạp tiền đồng thời lên tài khoản với số tiền balance
    - Chỉ được rút tiền khi số tiền trong tài khoản (balance) > số tiền cần rút.

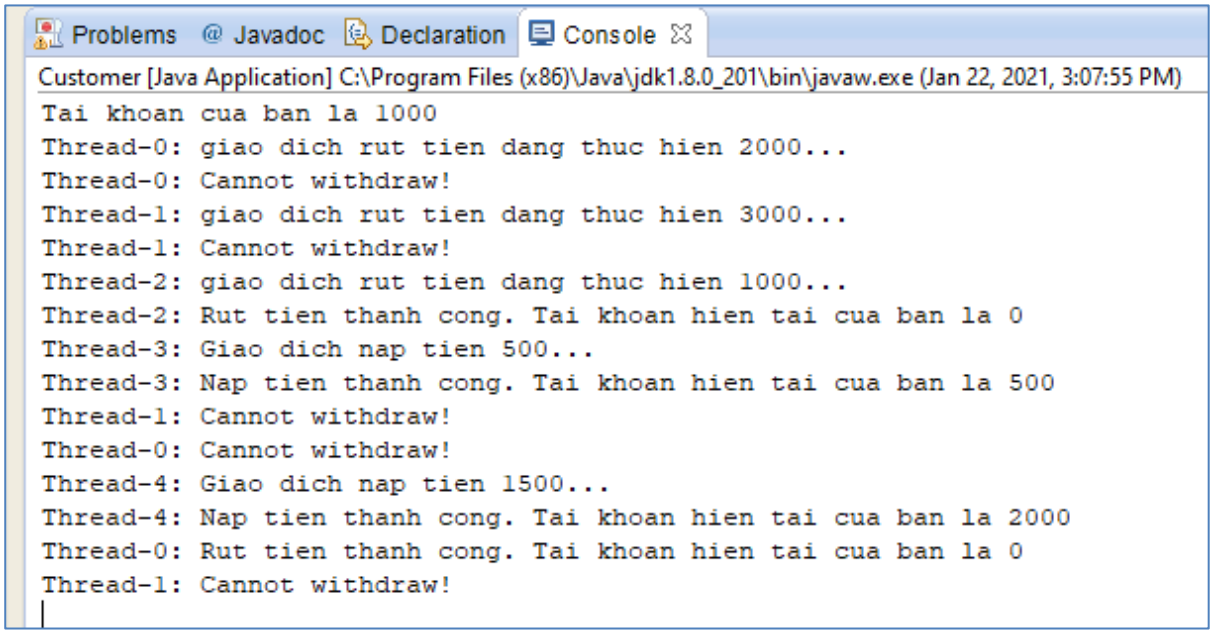
- Nếu có giao dịch nào đang thực hiện trên balance thì giao dịch khác phải chờ đến lượt.
- Giao dịch nào thực hiện xong phải trả lại balance và đánh thức giao dịch tiếp theo sử dụng balance.

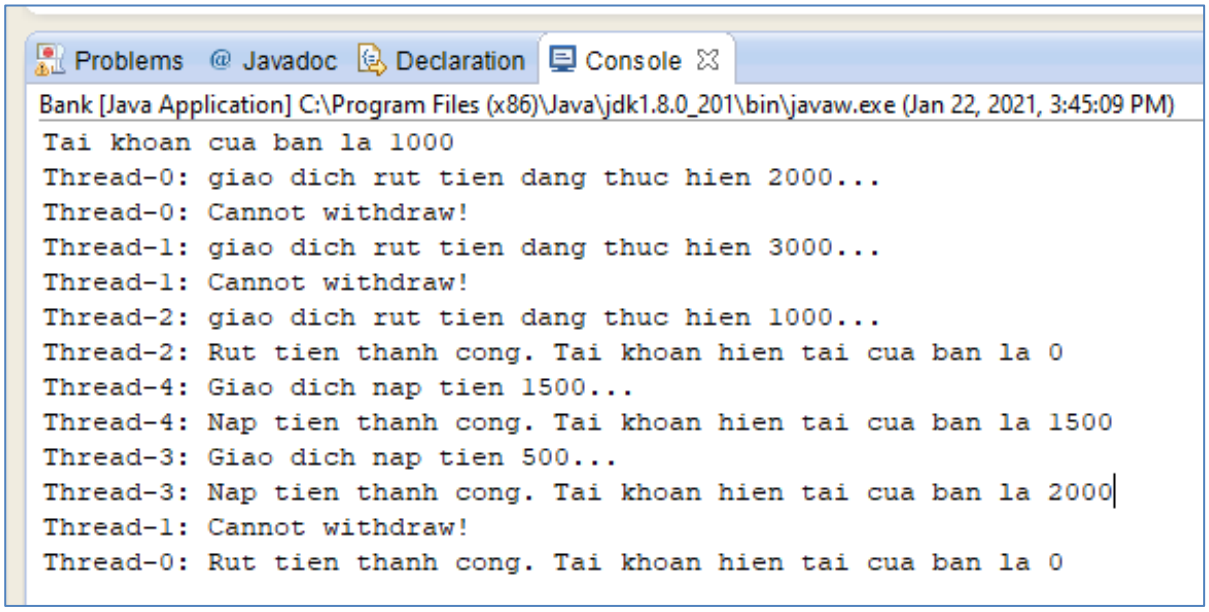
* Có thể mở rộng chương trìn máy rút ATM theo mô hình người sản xuất – người tiêu dùng

```
public class Bank {
   private int balance = 1000;
   public Bank() {
      System.out.println("Tai khoan cua ban la " + balance);
   }
   private synchronized void withdraw(int amount, String ten) {
         System.out.println(ten + ": giao dich rut tien dang thuc hien " + amount + "...");
      while(balance < amount) {
         System.out.println(ten + ": Cannot withdraw!");
         try {
            wait();
         } catch (InterruptedException ie) {
            System.out.println(ie.toString());
         }
      }
      balance -= amount;
      System.out.println(ten + ": Rut tien thanh cong. Tai khoan hien tai cua ban la " + balance);
   }
   private synchronized void deposit(int amount, String ten) {
      System.out.println(ten + ": Giao dich nap tien " + amount + "...");
      balance += amount;
      System.out.println(ten + ": Nap tien thanh cong. Tai khoan hien tai cua ban la " + balance);
      notifyAll();
     // notify();
   }
   public static void main(String[] args) {
      final Bank c = new Bank();
      Thread t1 = new Thread(){
        public void run() {
          // System.out.println(getName() +" : ");
           c.withdraw(2000,getName());
         }
      };
      t1.start();
      Thread t2 = new Thread(){
         public void run() {
                  //System.out.println(getName() +" : ");
                  c.withdraw(3000,getName());
         }
      };
      t2.start();
      Thread t3 = new Thread(){
         public void run() {
                  //System.out.println(getName() +" : ");
                  c.withdraw(1000,getName());
         }
      };
```

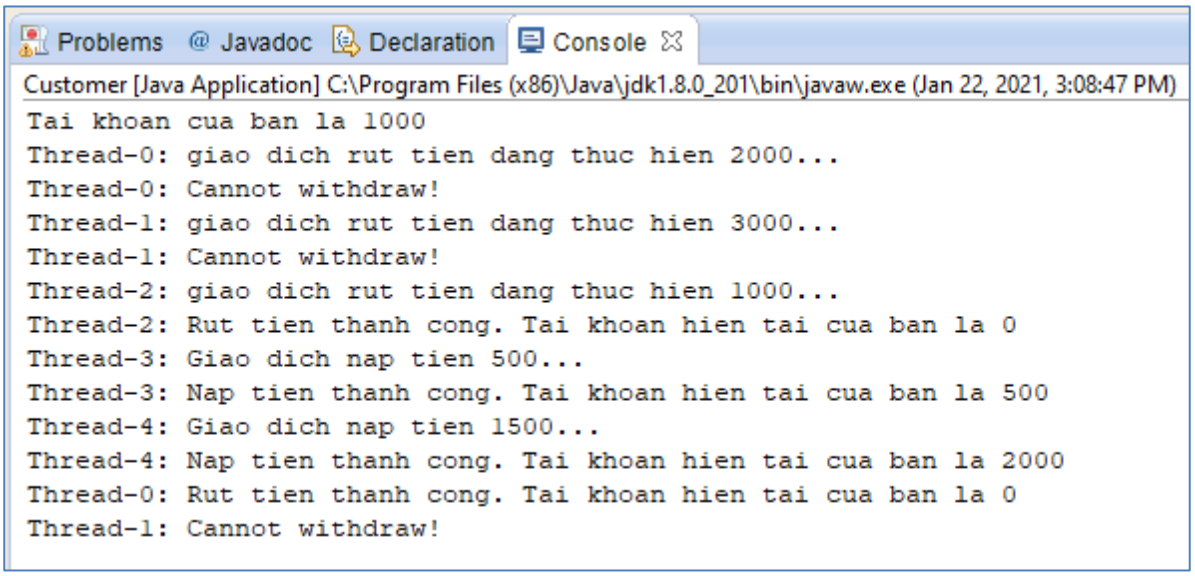
```
        t3.start();
        Thread t4 = new Thread(){
            public void run() {
            //        System.out.println(getName() +" : ");
                c.deposit(500, getName());
            }
        };
        t4.start();

        Thread t5 = new Thread(){
            public void run() {
            //        System.out.println(getName() +" : ");
                c.deposit(1500, getName());
            }
        };
        t5.start();
    }
}
```

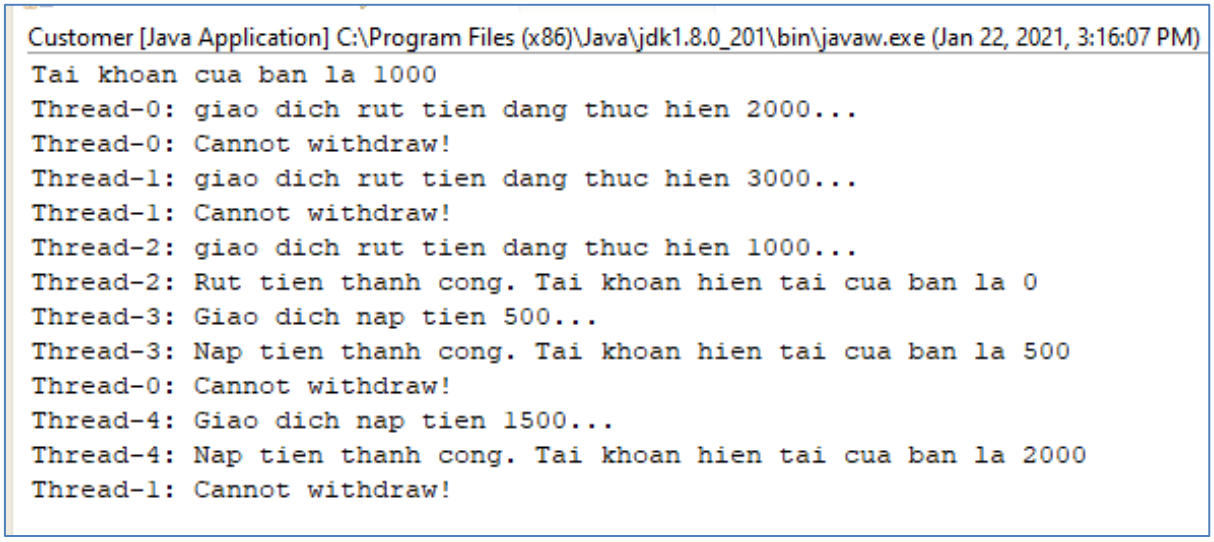
- Kết quả sử dụng phương thức notifyAll(): đánh thức tất cả các thread





- Kết quả sử dụng phương thức notify(): đánh thức ngẫu nhiên 1 thread





Bài 6: Tạo project có **Bai6.**.

- Viết chương trình thực hiện:
  - Thread thứ nhất thực hiện sinh ngẫu nhiên các số tự nhiêu từ 1 tới 20 - Dừng 2s và chạy vô tận
  - Thread thứ 2 thực hiện hiển thị bình phương các số được sinh ra từ Thread 1- Dừng 1s và chạy vô tận
  - Thực hiện đồng bộ 2 Thread trên (T1 thực hiện sinh ngẫu nhiên > T2 in ra bình phương > T1 tiếp tục sinh > T2 in ra bình phương)

Bài 7: Tạo project có tên là **Bai7**

- Sử dụng Thread xây dựng mô hình người sản xuất – người tiêu dùng

PRODUCER -------> QUEUE -------> CONSUMER

- Producer: liên tục tạo ra sản phẩm bỏ vào kho (Queue). Nếu kho đầy thì đợi.
- Consumer: kiểm tra sản phẩn có trong kho nếu có thì thực hiện lấy sản phẩm ra. Không có thì đợi.

```
1  public class BoundedBufferQueue {
2      private int arr[]; // mang luu tru san pham (kho)
3      private int max; //size cua mang
4      private static int lastPosition = 0;//luu vi tri san pham cuoi cung
5
6      /**
7       * contractor khoi tao khi new mot BoundedBufferQueue
8       */
9      public BoundedBufferQueue(int maxN) {
10         max = (maxN > 0)? maxN : 100;
11         arr = new int[max];//tao mang max phan tu
12     }
13
14     //them san pham vao kho
15     public synchronized void addProduct(int value){
16         try{
17             if(isFull()){//neu Kho day San pham
18                 Thread.currentThread().wait();//dung thread add SP
19             } else {
20                 arr[lastPosition] = value; //add sp vao Kho
21                 lastPosition++; //tang vi tri chua sp len 1
22                 System.out.println("Producer add: "+value +" ");
23             }
24            notifyAll();
25         }catch(Exception e){}
26     }
```

```
28      //lay san pham
29      public synchronized int getProduct(){
30           int result = Integer.MIN_VALUE; //san pham duoc ban
31           try{
32               if(isEmpty())              //neu mang trong
33                 Thread.currentThread().wait();  //thread lay SP doi
34               else if(lastPosition > 0){
35                   result = arr[0];  //lay san pham dau tien
36                   for(int i=0; i < lastPosition - 1; i++)
37                          arr[i] = arr[i+1]; //day cac sp phia sau len truoc
38                   lastPosition--;
39               }
40              notifyAll();
41           }catch(Exception e){}
42           return result;
43      }
```

```
44      public boolean isFull(){
45          return lastPosition == max;
46      }
47      public boolean isEmpty(){
48          return lastPosition == 0;
49      }
50      public static int getLastPosition() {
51          return lastPosition;
52      }
```

```
1  import java.util.Random;
2
3  public class Producer extends Thread{
4      private BoundedBufferQueue content; //khai bao kho
5
6      public Producer(BoundedBufferQueue content) {
7          this.content = content;
8      }
9      public void run()
10     {
11          while(true){
12              Random r = new Random();
13              int product = (int)(r.nextInt(100));//tao san pham tu 0->99
14              content.addProduct(product);
15          }
16     }
17 }
```

```java
public class Consumer extends Thread{

    private BoundedBufferQueue content; //khai bao kho

    public Consumer(BoundedBufferQueue content) {
        this.content = content;
    }

    public void run() {
        while(true){
            int x = content.getProduct(); //lay 1 sp ra khoi kho
            if(x != Integer.MIN_VALUE)
                System.out.println("Consumer get: " + x);
        }
    }
}
```

```java
public class Test {

    public static void main(String[] args) {
        BoundedBufferQueue kho = new BoundedBufferQueue(5);//tao kho 5 SP
        //tao ra 2 nguoi san xuat p và p2 (2 luồng sx)
        Producer p = new Producer(kho);
        Producer p2 = new Producer(kho);

        //tao ra 1 nguoi tieu dung (1 luồng td)
        Consumer c = new Consumer(kho);

        p.start();
        p2.start();
        c.start();
    }
}
```

Bài 8: Tạo project có tên là **Bai8**. Viết chương trình sau:

- Thread1: sinh ngẫu nhiên các số từ 1-100
- Nếu số ngẫu nhiên chia hết cho 3 thì đẩy sang thread2. Thread2 sẽ in ra bình phương của số đó.
- Nếu số ngẫu nhiên chia hết cho 2 thì đẩy sang thread3. Thread3 sẽ thực hiện kiểm tra và in thông báo số đó có chia hết cho 4 không
- Nếu tổng các số ngẫu nhiên trong Thread1 >=200 thì dừng tất cả các thread
- Thực hiện đồng bộ 3 thread theo thứ tự: Thread1 sinh số ngẫu nhiên xong → đợi cho thread2 và thread3 thực hiện xong mới in tiếp. Xong thì đánh thức thread1 và thread2 và thread3 phải chờ thread1 sinh số ngẫu nhiên xong mới làm tiếp.

Bài 9: Tạo project có tên là **Bai9**. Viết chương trình sau:

- Viết chương trình thực hiện:
  - Thread 1 thực hiện sinh ngẫu nhiên các ký tự a-z, cứ 2s thì sinh 1 lần
  - Thread 2 thực hiện biến các ký tự thường được sinh ra từ Thread 1 thành ký tự hoa, cứ sau 1s thì thực hiện 1 lần.
  - Yêu cầu đồng bộ 2 Thread (wait, notify)
  - Nếu số lần thực hiện của 2 thread là 5 lần thì dừng 2 thread lại.