HOT COLD

```
import java.util.Scanner;

class TooHot extends Exception {
  TooHot(String message) {
    super(message);
  }

  public String toString() {
    return "TooHotException: " + getMessage();
  }
}

class TooCold extends Exception {
  TooCold(String message) {
    super(message);
  }

  public String toString() {
    return "TooColdException: " + getMessage();
  }
}

class Temperature {
  int temp;
  Scanner sc = new Scanner(System.in);
  public void setTemp() {
    System.out.println("Enter the temperature: ");
    temp = sc.nextInt();
  }
  public void checkTemp() throws TooHot, TooCold {
    if (temp > 40) {
      throw new TooHot("Tempature is too hot");
    }
    else if(temp < 10) {
      throw new TooCold("Temperature is too cold");
    }
    else {
      System.out.println("Temperature is normal");
    }
  }
}

public class HotCold {
  public static void main(String[] args) {
    Temperature t = new Temperature();
    t.setTemp();
    try {
```

```
      t.checkTemp();
    } catch (TooHot e) {
      System.out.println(e);
    } catch (TooCold e) {
      System.out.println(e);
    }
  }
}
```

Two thread

```
import java.util.Scanner;

class MaxThread extends Thread {
  private int arr[];
  Scanner sc = new Scanner(System.in);

  public void setArray() {
    System.out.println("Enter the size of the array: ");
    int n = sc.nextInt();
    arr = new int[n];
    System.out.println("Enter the elements of the array: ");
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }
  }

  public void run() {
    int max = arr[0];
    for (int i = 1; i < arr.length; i++) {
      if (arr[i] > max) {
        max = arr[i];
      }
    }
    System.out.println("Max: " + max);
  }
}

class MaxThreadUsingRunnable implements Runnable {
  private int arr[];
  Scanner sc = new Scanner(System.in);

  public void setArray() {
    System.out.println("Enter the size of the array: ");
    int n = sc.nextInt();
    arr = new int[n];
    System.out.println("Enter the elements of the array: ");
```

```
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }
  }

  public void run() {
    int max = arr[0];
    for (int i = 1; i < arr.length; i++) {
      if (arr[i] > max) {
        max = arr[i];
      }
    }
    System.out.println("Max: " + max);
  }
}

public class TwoThread {
  public static void main(String[] args) {
    MaxThread t = new MaxThread();
    MaxThreadUsingRunnable r = new MaxThreadUsingRunnable();
    Thread t1 = new Thread(r);
    t.setArray();
    r.setArray();
    t.start();
    t1.start();
  }
}
```

ProducerConsumer

```
class Utility {
  int n;
  boolean valueSet = false;

  synchronized int get() throws InterruptedException {
    while (valueSet) {
      System.out.println("Got: " + n);
      valueSet = false;
      notify();
    }
    wait();
    return n;
  }

  synchronized void put(int n) throws InterruptedException {
    if (!valueSet) {
      this.n = n;
```

```
      System.out.println("Put: " + n);
      valueSet = true;
      notify();
    }
    wait();
  }
}

class Producer implements Runnable {
  Utility utility;

  Producer(Utility utility) {
    this.utility = utility;
    new Thread(this, "Producer").start();
  }

  public void run() {
    int i = 0;
    while (true) {
      try {
        Thread.sleep(1000);
        utility.put(i++);
      } catch (InterruptedException e) {
        System.out.println("InterruptedException caught");
      }
    }
  }
}

class Consumer implements Runnable {
  Utility utility;

  Consumer(Utility utility) {
    this.utility = utility;
    new Thread(this, "Consumer").start();
  }

  public void run() {
    while (true) {
      try {
        Thread.sleep(1000);
        utility.get();
      } catch (InterruptedException e) {
        System.out.println("InterruptedException caught");
      }
    }
  }
}
```

```
public class ProducerConsumer {
  public static void main(String[] args) {
    Utility utility = new Utility();
    new Producer(utility);
    new Consumer(utility);
    System.out.println("Press Ctrl+C to stop.");
  }
}
```