Suppose that we have A, which is a two-dimensional array of int. Which of the following gives us the item in A which is stored in the last row, first column?

- A[0][A[0].length-1]
- A[length][0]
- A[A.length-1][0]
- A[0].length-1
- Impossible to tell with the information given.

Suppose that we have A, which is a two-dimensional array of int. Which of the following gives us the item in A which is stored in the last row, first column?

- A[0][A[0].length-1]
- A[length][0]
- *** A[A.length-1][0] ***
- A[0].length-1
- Impossible to tell with the information given.
What’s Printed?

```java
public class WhatsPrinted {
    public static void func(int A[], int x) {
        for (int i=0; i<A.length; i++)
            A[i]+=x;
    }

    public static void main(String args[]) {
        int []A = new int[3];
        A[0]=55;
        A[1]=22;
        func(A, A[1]);
        System.out.println(A[0]);
    }
}
```

Answer: 77

What’s Printed?

```java
public class WhatsPrinted {
    public static void mystery(int A[], int x) {
        for (int i=1; i<A.length; i++) {
            if (i%2==0) {
            } else {
                A[i]-=A[i-1]+x;
            }
        }
    }

    public static void main(String args[]) {
        int []A = {10,20,30,40};
        mystery(A, 2);
        System.out.println(A[A.length-1]);
    }
}
```

Answer: -2
public class WhatsPrinted {
    public static void mystery(int y, int z) {
        if (y+3 > z) {
            y++;
        } else {
            y--;
        }
    }
    public static void main(String args[]) {
        int x=10;
        int y=20;
        int z=30;
        mystery(x, y);
        System.out.println(y);
    }
}

Answer: 20
What's Printed?

```java
class Stuff {
    int x;
    /* constructor */
    public Stuff(int newX) {
        x = newX;
    }
}

public class WhatsPrinted {
    public static void func(Stuff s) {
        s.x *= 2;
    }
    public static void main(String args[]) {
        Stuff s1 = new Stuff(10);
        Stuff s2 = new Stuff(4);
        func(s2);
        s2.x += 2;
        if (s1 == s2) {
            System.out.println("same");
        } else {
            System.out.println("different");
        }
    }
}
```

Answer: different

Which analogy is most accurate?
- cookie cutter is to cookie as object is to class
- cookie is to cookie cutter as object is to class
- cookie cutter is to cookie as blueprint is to object
- cookie is to cookie cutter as blueprint-class is to class
- cookie is to cookie cutter as base class is to object

What's Printed?

```java
class Stuff {
    int x;
    /* constructor */
    public Stuff(int newX) {
        x = newX;
    }
}

public class WhatsPrinted {
    public static void main(String args[]) {
        Stuff s1 = new Stuff(10);
        Stuff s2 = new Stuff(4);
        func(s2);
        s2.x += 2;
        if (s1 == s2) {
            System.out.println("same");
        } else {
            System.out.println("different");
        }
    }
}
```

Which analogy is most accurate?
- cookie cutter is to cookie as object is to class
- *** cookie is to cookie cutter as object is to class ***
- cookie cutter is to cookie as blueprint is to object
- cookie is to cookie cutter as blueprint-class is to class
- cookie is to cookie cutter as base class is to object
What's the biggest difference in the way static and instance methods are used?

- Static methods operate on constants. Instance methods do not.
- If a method is static, it cannot be changed in a child class.
- A static method cannot change any arguments passed to the method. Instance methods can.
- The result of an instance method is volatile (i.e., it must be used in an instant). The result of static methods remain throughout the program.
- An instance method will operate on a particular object of a class. Static methods do not.

What is printed by the following?

```java
class Stuff {
    int x;
    public Stuff(int x) {
        this.x = x;
    }
    public void inc() {
        x++;
    }
    public static void main(String args[]) {
        Stuff s1 = new Stuff(10);
        Stuff s2 = s1;
        s1.inc();
        func(s1);
        System.out.println(s2.x);
    }
}

class WhatsPrinted07 {
    public static void func(Stuff s) {
        s.inc();
    }
    public static void main(String args[]) {
        Stuff s1 = new Stuff(10);
        Stuff s2 = s1;
        s1.inc();
        func(s1);
        System.out.println(s2.x);
    }
}
```

Answer: 12
public class SweDihShh {
    public static void main(String args[]) {
        String s1="swe";
        String s2="dih";
        String s3="shh";

        for (int i=0; i<5; i++) {
            if (i%2==0) {
                s2+=s3;
            } else {
                s1+=s2;
            }
        }
        System.out.println(s1);
    }
}

Which of the following methods returns the location of the largest element in A? You may assume that A contains at least one element.

```java
public static int indexOfLargest(int []A) {
    int L=A[0];
    for (int i=1; i<A.length; i++) {
        if (A[i]>A[L]) {
            L=A[i];
        }
    }
    return L;
}
```

Which of the following methods returns the location of the largest element in A? You may assume that A contains at least one element.

```java
public static int indexOfLargest(int []A) {
    int L=A[0];
    for (int i=1; i<A.length; i++) {
        if (A[i]>A[L]) {
            L=A[i];
        }
    }
    return L;
}
```
Which of the following methods returns the location of the largest element in A? You may assume that A contains at least one element.

```java
public static int indexOfLargest(int []A) {
    int L=A[0];
    for (int i=1; i<A.length; i++) {
        if (A[i]>L) {
            return i;
        }
    }
    return L;
}
```

Nope
Which of the following methods returns the location of the largest element in A? You may assume that A contains at least one element.

```java
public static int indexOfLargest(int[] A) {
    int L = 0;
    for (int i = 1; i < A.length; i++) {
        if (A[i] > A[i - 1]) {
            L = i;
        }
    }
    return L;
}
```

Yes!
What is printed by the following?

```java
public class SD1 {
    public static void main(String []args) {
        int []A = new int[3];
        int []B;


        B = A;

        if (A == B) {
            System.out.println("same");
        } else {
            System.out.println("different");
        }
    }
}
```

Answer: same

What is printed by the following?

```java
public class SD2 {
    public static void main(String []args) {
        int []A = new int[3];
        int []B = new int[3];


        for (int i = 0; i < A.length; i++) {
            B[i] = A[i];
        }

        if (A == B) {
            System.out.println("same");
        } else {
            System.out.println("different");
        }
    }
}
```

Answer: different
public class SD3 {
    public static int[] f(int[] A, int x) {
        int[] B = new int[A.length];
        for (int i = 0; i < A.length; i++) {
            B[i] = A[i] * x;
        }
        return B;
    }
    public static void main(String[] args) {
        int[] A = {15, 30, 45};
        int[] B = {5, 10, 15};
        int[] C = f(B, 3);
        if (A == C) {
            System.out.println("same");
        } else {
            System.out.println("different");
        }
    }
}

Answer: different

Some Inheritance and Polymorphism Questions
Determine Hierarchy

Class Definitions
- Stopen extends Girsby
- Finfin extends Blavik
- class Girsby
- Blavik extends Girsby

Visually

Legal?

Stopen s = new Girsby();

Answer: No
Blavik b = new Blavik();
Girsby g = b;
Blavik ab = g;

Girsby [] theGirsbies = {
    new Stopen(),
    new Blavik(),
    new Girsby(),
    new Finfin()
};
Girsby g = new Stopen();
System.out.println(g.f3());

Girsby f = new Finfin();
System.out.println(f);
Write a function `insert()`, which is passed an `int n`, an `int A[]`, and an `int pos` (for “position”). The function inserts `n` into the array that `A` references at `A[pos]` shifting all the items in the array that had been at index `pos` or greater one place to the right. For example, if the array is

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

and we call `insert(A, 15, 1)`, the array `A` references then becomes:

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>5</td>
<td>15</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

the element in the array that’s in the right-most position is removed. If `pos < 0` or `pos >= A.length-1`, the method does nothing.

Write a method which is passed the name of a file. The method returns the word in the file that has the greatest number of vowels. Recall that the vowels are the letters {‘a’, ‘e’, ‘i’, ‘o’, and ‘u’} and the capital letter equivalents. You may assume that the file contains nothing but words and inter-word space (i.e., no punctuation or letters). If no word contains a vowel, return an empty String (“”). You do not have to handle `FileNotFoundException`.

One Possibility

```java
public static void insert(int A[], int n, int index) {
    if (index < 0 || index >= A.length) {
        return;
    }
    for (int i = A.length - 1; i > index; i--) {
        A[i] = A[i - 1];
    }
    A[index] = n;
}
```

One Possibility

```java
public static String mostVowels(String filename)
    throws FileNotFoundException {
    String mostSoFar = "";
    Scanner in = new Scanner(new File(filename));
    while (in.hasNext()) {
        String cur = in.next();
        if (numVowels(cur) > numVowels(mostSoFar)) {
            mostSoFar = cur;
        }
    }
    return mostSoFar;
}
```
One Possibility

public static String mostVowels(String filename)
    throws FileNotFoundException {
    String mostSoFar = "";
    Scanner in = new Scanner(new File(filename));
    while (in.hasNext()) {
        String cur = in.next();
        if (numVowels(cur) > numVowels(mostSoFar)) {
            mostSoFar = cur;
        }
    }
    return mostSoFar;
}

Is there a numVowels() function?
No, but:
▶ Not hard to write
▶ Makes life easier, simpler to read, harder to screw up

One Possibility

public static int numVowels(String s) {
    int count = 0;
    for (int i=0; i<s.length(); i++) {
        if (isVowel(s.charAt(i))) {
            count++;
        }
    }
    return count;
}

Is there an isVowel() function?
No, but:
▶ Not hard to write
▶ Makes life easier, simpler to read, harder to screw up

One Possibility

public static boolean isVowel(char c) {
    String vowels = "aeiouAEIOU";
    for (int i=0; i<vowels.length(); i++) {
        if (c == vowels.charAt(i)) {
            return true;
        }
    }
    return false;
}
One Possibility

```java
public static boolean isVowel(char c) {
    String vowels = "aeiouAEIOU";
    for (int i=0; i<vowels.length(); i++) {
        if (c == vowels.charAt(i)) {
            return true;
        }
    }
    return false;
}
```

Lots of different correct ways

- Challenge in programming: break big problem into small parts
- Functions help you. Please use them

Shoe class

Create a Shoe class.

- Provide instance fields for brand and color which are of type String, and size, which is a double. Each of these should be accessible only within the Shoe class.
- Create a constructor, which initializes the instance fields
- Write a toString() method, which returns a String representation of the Shoe.
- Write an equals() method. Two Shoes are the same if they have the same brand, color, and size.

Many correct answers. Here’s one

```java
public class Shoe {
    private String brand;
    private String color;
    private double size;

    public Shoe(String b, String c, double s) {
        brand = b;
        color = c;
        size = s;
    }

    public String toString() {
        return brand + " size " + s;
    }

    public boolean equals(Shoe o) {
        return brand.equals(o.brand) &&
            color.equals(o.color) &&
            size == o.size;
    }
}
```