Week 11 Tutorial

Brae

May 18, 2018

CSSE2002: Programming in the Large

This Week

This week we will be working on questions from the 2017 `Semester Two Final Exam Paper`.

1. Go to the UQ Library Website (library.uq.edu.au)
2. Select `Past exam papers` from the search dropdown
3. Enter `CSSE2002` into the search field
4. Select `2017 Sem 2`

Exclude the following questions:

1. 1b
2. 2a
3. 3
4. 4b

Question One

Make the below a declaration of a constant:

```
int maximum = 100;
```

What is printed out by this program?

```
public static void absArray(int [ ] a) {
    for (int i = 0; i < a.length; i++)
        if (a[i] < 0)
            a[i] = -1 * a[i];
    }
public static void main(String [ ] args) {
    int [ ] values = {1, 3, -2, 0, -10, 9};
    absArray(values);
    for (int i = 0; i < values.length; i++)
        System.out.println("values[" + i + "] = " +
            values[i]);
}
```

Question One

Make the below a declaration of a constant:

```
int maximum = 100;
```

```
static final int MAXIMUM = 100;
```

Question One

What is printed out by this program?

```
public static void absArray(int [ ] a) {
    for (int i = 0; i < a.length; i++)
        if (a[i] < 0)
            a[i] = -1 * a[i];
    }
public static void main(String [ ] args) {
    int [ ] values = {1, 3, -2, 0, -10, 9};
    absArray(values);
    for (int i = 0; i < values.length; i++)
        System.out.println("values[" + i + "] = " +
            values[i]);
}
```

Question One

What is printed out by this program?

```
public static void absArray(int [ ] a) {
    for (int i = 0; i < a.length; i++)
        if (a[i] < 0)
            a[i] = -1 * a[i];
    }
public static void main(String [ ] args) {
    int [ ] values = {1, 3, -2, 0, -10, 9};
    absArray(values);
    for (int i = 0; i < values.length; i++)
        System.out.println("values[" + i + "] = " +
            values[i]);
}
```

values[0] = 1 values[1] = 3 values[2] = 2 values[3] = 0 values[4] = 10 values[5] = 9

Question One

```
public static void r(int[] a) {
    int[] b = a;
    r(b, 0, b.length - 1);
}
public static void r(int[] a, int i, int j) {
    if (i <= j) {
        int tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;
        r(a, i + 1, j - 1);
    }
}
public static void main(String[] args) {
    int[] values = {1, 3, -2, 0, -10, 9};
    r(values);
    for (int i = 0; i < values.length; i++)
        System.out.println("values[" + i + "] = " +
            values[i]);
```

Question One

values[0] = 9

values[1] = -10

values[2] = 0

values[3] = -2

values[4] = 3

values[5] = 1

Question Two

```
public class Person {
    private String name;
    public Person(String n) { name = n; }
    public String getName() { return name; }
    public void setName(String n) { name = n ; }
}
```

The class invariant for this class is indented to be that only legal characters are used in a persons name, as defined by the `legalNameCharacter` method below.

```
public static boolean legalNameCharacter(char c) {
    return "a" <= c && c <= "z" k ||
           "A" <= c && c <= "Z" k ||
           c == "-" k || c == " ";
}
public class IllegalNameException extends Exception {
    }
```

Question Two

```
public class Person {
    private String name;
    public Person(String n) {
        setName(n);
    }
    public String getName() {
        return name;
    }
    public void setName(String n) {
        if (!isValidName(n)) {
            throw new IllegalNameException();
        }
        name = n ;
    }
    private boolean isValidName(String name) {
        for (char letter : name.toCharArray()) {
            if (!legalNameCharacter(letter)) {
                return false;
```

Question Two

```
    private boolean isValidName(String name) {
        for (char letter : name.toCharArray()) {
            if (!legalNameCharacter(letter)) {
                return false;
            }
        }
        return true;
    }
}
```

Question Four

```
public class A {
    public int m()
    public int m(int x, int y)
}

public class B extends A {
    public int m(double x, double y)
}

public class C extends B {
    public int m()
    public int m(int x, double y)
}
```

1. b.m();
2. c.m(1, 2.1);
3. b.m(1, 2.1);
4. b.m(1, 2);
5. c.m(1.1, 2.1);
6. c.m(1, 2);

Question Four

The second call is tricky. You might find the below description useful.
When a method is invoked (15.12), the number of actual arguments (and any explicit type arguments) and the compile-time types of the arguments are used, at compile time, to determine the signature of the method that will be invoked (15.12.2). If the method that is to be invoked is an instance method, the actual method to be invoked will be determined at run time, using dynamic method lookup (15.12.4).

Question Four

1. A.m()
2. B.m(double, double)
3. B.m(double, double)
4. A.m(int, int)
5. B.m(double, double)
6. A.m(int, int)