Functions *(cont.)*

CIS 110 July 13 Recitation

Method Overloading

- Having two or more methods in the same class (in our case, the same .java file) with the same name and return type but different parameters.
- Example: System.out.println(...);
  - To understand this example, just note that “println” is a method with return type void in a Java library called the PrintStream class. The method prints the input and terminates the line.
  - void println(boolean x);
  - void println(char x);
  - void println(double x);
  - void println(int x);
  - void println(String x);

Review: Why do we need functions (methods)?

- Break code down into logical sub-steps.
  - Methods should be cohesive: compact enough to represent a single abstract step or calculation.
  - Methods should be general enough that they can be reused many times.
- Improve readability of code (for others and for yourself).
- Testability: methods allow the programmer to focus on correctly implementing each individual logical sub-step of a problem.
  - Minimize unwanted side effects!

Testing: Minimize bugs

- Compile and run your program frequently to make sure your program is still running the way you expect it to.
- Write the skeleton of your methods (method signature and corresponding return statements) first so that you can compile your program.
- Check your program by writing test cases inside your main method before implementing your functions.
    - Compare your expected value (computed by hand) with actual values (method outputs).
    - As a general rule, write as many test cases as are needed to consider edge cases.
    - For if-else statements, test every branch.

```java
public class MyClass {

    public static int findMax(int[] array) {
        int max = Integer.MIN_VALUE;
        // TODO: fill this in later
        return max;
    }

    public static double average(int[] array) {
        double avg = 0.0;
        // TODO: fill in later
        return avg;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6, 7};
        System.out.println("Testing findMax:");
        System.out.println("Expected : " + 7);
        System.out.println("Actual : " + findMax(arr));

        System.out.println("Testing average:");
        System.out.println("Expected : " + 4.0);
        System.out.println("Actual : " + average(arr));
    }
}
```

Coding Exercise

- **public static int numElements (int[] array)**
  - Return the number of elements in “array”
- **public static boolean isEmpty (int[] array)**
  - Return true if “array” has no elements, false otherwise.
- **public static String sortHouse (String name)**
  - Return name of house “name” would be sorted into using the following rules:
    - “name” is less than 3 letters: Gryffindor
    - “name” is 3 or more letters but less than or equal to 7: Ravenclaw
    - “name” is more than 7 letters but less than 10: Slytherin
    - Everyone else belongs in Hufflepuff
- **public static String reverseString (String input)**
  - Return the reversed version of “String”
- Remember to write your test cases **first** in the main method.