Week 1

Exercise 1.1.5

```java
public class UseArgument {
  public static void main(String[] args) {
    System.out.print("Hi, ");
    System.out.print(args[0]);
    System.out.println(". How are you?");
  }
}
```

Listing 1: UseArgument

Describe what happens if you try to execute `UseArgument` with each of the following command lines:

a. `java UseArgument java`

   Prints `java` as the name: `Hi, java. How are you?`

b. `java UseArgument @!&^%`

   This will return different results depending on the shell you're using. If you pass those arguments directly as-is, it will just print `Hi, @!&^%. How are you?`.

   If your shell uses any of these characters, not all arguments may be passed to the java program, you may get an error, or get no output. For example, in a POSIX-compatible shell like bash, `&` is used as a delimiter to run programs asynchronously (return user control before the program finishes running), and otherwise acts as `;`. An error may be thrown due to `^%` not being a known command.

c. `java UseArgument 1234`

   Prints `Hi, 1234. How are you?`

d. `java UseArgument.java Bob`

   Newer Java versions run `javac` if a `.java` file is provided as the class name to run, and then run the compiled bytecode.

e. `java UseArgument Alice Bob`

   Arguments are whitespace delimited (in most shells), and only the first one is read in the program: `Hi, Alice. How are you?`

Exercise 1.1.6

Modify `UseArgument.java` to make a program `UseThree.java` that takes three names as command-line arguments and prints a proper sentence with the names in the rev

```java
public class UseThree {
  public static void main(String[] args) {
    if (args.length != 3) {
      System.out.println("Error: Program expects exactly three arguments");
      return;
    }
    System.out.printf("Hi %s, %s and %s. How are you?%n", args[2], args[1], args[0]);
  }
}
```

Listing 2: UseThree