Garfield AP CS

comments, methods, flow control

Warm-up

- What is the output of the following `println` statements?

```
System.out.println("\ta\tb\tc");
System.out.println("\\\\");
System.out.println("'");
System.out.println("\"\"\"");
System.out.println("C:\nin\the downward spiral");
```

- Write a `println` statement to produce this output:

```
/ \ // \\ /// \\\
```

“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.”

M. Golding

Comments

- Lets others know what's on your mind
- Explains tricky bits
- Must be used sparingly
- /* */ and //

Do it like this:

```
/* Prints a greeting */
```

Not like this:

```
/* This is my super awesome
program that uses the
println statement to go
ahead and display a friendly
message to the user because
it's a convention that was
started long ago.*/
```

Algorithms

- **algorithm**: A list of steps for solving a problem.
- Example algorithm: "Bake sugar cookies"
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven temperature.
  - Set the timer.
  - Place the cookies into the oven.
  - Allow the cookies to bake.
  - Spread frosting and sprinkles onto the cookies.
  - ...

What are some potential problems?

Structured algorithms

- **structured algorithm**: Split into coherent tasks.

  1 Make the cookie batter.
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.

  2 Bake the cookies.
  - Set the oven temperature.
  - Set the timer.
  - Place the cookies into the oven.
  - Allow the cookies to bake.

  3 Add frosting and sprinkles.
  - Mix the ingredients for the frosting.
  - Spread frosting and sprinkles onto the cookies.

  ...

Static methods

- **static method**: A named group of statements.
  - denotes the *structure* of a program
  - eliminates *redundancy* by code reuse

  - **procedural decomposition**:
    dividing a problem into methods

- Writing a static method is like adding a new command to Java.

Decomposition

- Decide what your related steps are
- Group the steps in a method
- Name the method descriptively
- Call your new method

```
public static void main(String[] args) {
  chorus();
}

public static void chorus() {
  System.out.println("P-p-p-Poker Face, P-p-p-Poker Face");
  System.out.println("Mum mum mum mah");
}
```

When to use methods

- Statements are closely related
- Statements are repeated
- Watch out for weakly-related statements
- You can always change your decomposition!

Methods calling methods

Output of the following program?

```
/* Helene Martin, Garfield AP CS, 2009
   Demonstration of methods calling methods  */
public class FlowControl {
   public static void methodOne() {
      System.out.println("foo");
      methodThree();
   }
   public static void methodTwo() {
      System.out.println("bar");
      methodOne();
   }
   public static void methodThree() {
      System.out.println("baz");
   }
   public static void main(String[] args) {
      methodOne();
      methodThree();
      methodTwo();
   }
}
```

Control flow

- Sequential, starting with statement in main
- Method calls make computer "jump" to method statements

- Then it comes back to where it was
- Use debugger to know where you are
- Call stack shows where you are and where you'll go back to

Exercise

- Write a program to print these figures using methods.