

Working with Classes

Atul Prakash

Functions

- In Python and C++, you are used to simply using functions.
- In Java, functions can be created by prefixing the function with the word "static".
- Non-static: methods that apply to a specific object instance

Example

```
double root = Math.sqrt (17.0);  
double angle = 1.5;  
double height = Math.sin (angle);
```

sqrt

```
public static double sqrt(double a)
```

Returns the correctly rounded positive square root of a `double` value. Special cases:

- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the `double` value closest to the true mathematical square root of the argument value.

Parameters:

a - a value.

Returns:

the positive square root of a. If the argument is NaN or less than zero, the result is NaN.

Code for the Math class

<http://www.docjar.com/html/api/java/lang/Math.java.html>

Or google search for "Math java source"

Static Variables

- All variables must be declared within a class.
- Static variables: class-specific
- Non-static variables: object-specific

Example

- `double x = Math.PI;`
- Declaration of PI within the Math class:
 - `public static final double PI =
3.14159265358979323846;`

Constant variables

- Variables can be declared as "final" to tell Java that they will not change in value.
 - `public static final double PI = 3.14159265358979323846;`

Public versus Private

- **private:** visible only within the class
 - `private String myname;`
- **public:** visible outside the class and package
 - `public String getName()`

Design Tradeoffs

```
public class Bicycle {  
    public int gear;  
    public int maxspeed;  
    public static int numberOfBicycles = 0;  
  
    public Bicycle(int g, int s) {  
        gear = g;  
        maxspeed = s;  
        numberOfBicycles++;  
    }  
}
```

Which variables are object-specific, which ones are global to the whole class?

Should gear and maxspeed be public variables?

If they are public:

- Following is possible:
 - `Bicycle b = new Bicycle(10, 60);`
 - `b.gear = -2; // should be illegal`
 - `b.maxspeed = -10; // should be illegal`
- May be it is a good idea to do some error checking when setting gear and maxspeed?

Setters and Getters

```
public class Bicycle {
    private int gear;
    private int maxspeed;
    public static int numberOfBicycles = 0;

    public Bicycle(int g, int s) {
        gear = g;
        maxspeed = s;
        numberOfBicycles++;
    }

    public boolean setSpeed(int s) {
        if (s < 0 || s > 150) {
            return false;
        }
        maxspeed = s;
        return true;
    }

    public int getSpeed() {
        return maxspeed;
    }

    public boolean setGear(int g) {
        if (g < 0) return false;
        gear = g;
        return true;
    }

    public int getGear() {
        return gear;
    }
} // end class
```

gear and *maxspeed* made private. Added setters and getters. (Eclipse can generate setters and getters.)

Even better...

- Make `numberOfBicycles` private. It tracks the number of bikes created.
- Add a getter for `numberOfBicycles`
- The getter will be static because it is a class function -- not specific to a particular bike.

```
public class Bicycle {
    private int gear;
    private int maxspeed;
    private static int numberOfBicycles;

    public static int getNumberOfBicycles() {
        return numberOfBicycles;
    }

    public Bicycle(int g, int s) {
        gear = g;
        maxspeed = s;
        numberOfBicycles++;
    }
    ... rest same as before ...
}
```

GameHelper class

- Re-examine the GameHelper class.
- Can the getUserInput() method be made static?
- Pros and cons?
- If you make it static, what change should be made in the SpaceshipGame.main()? Is the change optional or required?
- Try making the change. Report your experience on ctools (mini-exercise).

main program in Java

- To run a Java program:
`java <Classname>`
- Java executes
`<Classname>.main()`
method.
 - Must be declared
`static` since it is class-specific, not object-specific
- When the method completes, the program ends.
- It is OK for multiple classes to have `main()` methods. E.g.,
`SpaceshipGame` and
`SpaceshipGameTest`.
Only one is executed.

