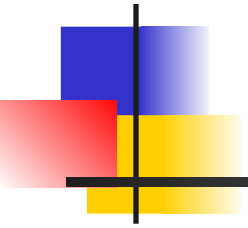

Java Program Structure

Introduce basic Java program structure.

James Brucker

Structure of Java source code

```
import java.util.Scanner;
import java.time.LocalDate;
/**
   *  A simple class.
   *  @author Bill Gates
   */
public class Greeter {




}
// ERROR - don't write code here
```

1. "import" other classes

2. Javadoc comment.

Start of the class

End of the class

{ } define a "scope"

```
public class Greeter {


   public static void main( String [] args)
   {
                In Java, { ... } defines a block
                of code.  You will see { }
                blocks used in many ways.


    }
}
```

The { and } braces indicate the start and end of the code for this class.

Define a method

```
public class Greeter {


   public static void methodName( ) {

          statement1;
          statement2;
          statement3;

    }


}
```

method name starts with lowercase

scope of method is defined by { ... }

main method

```
public class Greeter {


public static void main( String[ ] args ) {
        System.out.print("Who are you?");
        statement2;
        statement3;

    }


}
```

to execute a class as a program, it must have a main method **exactly like this**

Local Variables

```
public class Greeter {

   public static void sayHello( ) {
        String greet = "Hello.";
        System.out.println( greet );
        int counter = 0;
        . . .
    }
    public static void main(String[] a) {
        sayHello( );
  // error: counter is not defined here
        System.out.println(counter);
}
```

Local variables are declared *inside* a method. They only exist while method is executing.

class can have many methods

```
public class Greeter {
    public static void method1( ) {
        System.out.print("This is method1");
    }

    public static void method2( ) {
        System.out.print("This is method2");
    }


    public static void main( String[ ] args ) {
        method1( );
        method2( );
        method1( ); // call method1 again
    }
}
```

a class can have many methods, using different names.

3 Kinds of Comments

```
/**
 * Javadoc comment describes this class.
 */
public class Greeter {
   /*
    A multi-line comment can be
    very long.
    */
    public static void method1( ) {
        // a single line comment
        System.out.print("This is method1");
        int n = 0;  // another comment
    }
}
```

The compiler ignores comments.

Javadoc comments create online documentation for your code.

(static) attributes

```
public class Greeter {
   static String greet = "Hello.";

   public static void sayHello( ) {

        System.out.println( greet );
        . . .
   }


}
```

A variable defined outside of a method is an attribute of the class.

Static attributes can be used in any method, but usually in static methods.

Saving the Program

```
/** Print an impersonal greeting message
 *  @author James Brucker
 */
public class Greeter {
   static String greeting = "Hello, ";
   /* execution starts in the main method
   The header (signature) line must
   be as shown here.
    */
   public static void main( String [] args) {
     String who = "Human";
     // print a message on terminal
     System.out.println(greeting + who);
     System.out.println( "Goodbye, "
   + who );
   }
}
```

Filename: **`Greeter.java`**

The name of the file *must* be exactly the same as the name of the class in the file, with an Extension "`.java`"

Filename: **`Bank.java`**

public class Bank { ... }

WRONG:

Filename: **`bank.java`**

public class Banking { ... }

General Program Structure

```
package greeting;
import java.util.Scanner;
import java.time.LocalTime;
/** Print an impersonal greeting message
 *  @author James Brucker
 */
public class Greeting  {
    public static final Strng GREET = "Hello";
    private static int counter = 0;
    /** instance variable */
    private String name;
    /** constructor for new objects
     * @param name is person to greet
     */
    public Greeting ( String name )  {
        this.name = name;
    }
    public void greet( ) {
        System.out.println(GREET + name);
```

1. package name (optional)
2. import statement(s) - may have many.
3. Javadoc comment for class
4. Start of the class

Contents of Class:

1. define constants first
2. static variables
3. instance variables
4. constructor(s) - optional
5. methods
6. private methods

method names: camelCase

Review

- In Java, all code must be part of a class.
- A class begins with the declaration:

  `public class SomeClassName`

  followed by the class definition inside { ... }
- "public" means that this class is visible to other classes.
- Inside a class, code is contained in *methods*.
- A method definition is delimited by { ... }
- This main method is where program execution begins. The main method must have this header line:

```
public static void main( String [ ] args )
```

Review

- Inside a method we can define local variables and assign values to them.
- To define a String variable and assign a value, use:

  String *variableName* = "*some value*" ;

- To display a String on the console, we use:

  System.out.println("Are you awake?");

A Thought Question

```
System.out.println( "What does this mean?" );
```

Q: What is System ?

Q: What is "out" in System.out ?

Q: What is "println" in System.out.println(...) ?