

Methods and Classes

Functions in Java

- In Java, every program is a “class”.
 - i.e. object type
 - There are no functions outside of a class.
- All functions are inside a class.
 - thus called “methods”
 - eg. the `main()` method we have used so far:

```
public static void main(String[] args) { ... }
```
 - ... it’s inside the class we’re creating.

Basics

- Syntax:

`<method defn> ::= <modifiers> <type> <identifier>
(<parameters>) { <method body> }`

- Example:

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

- `main` is the method identifier.
- The `{ ... }` is the method body.
 - code that runs when it's called

Parameters

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

- The arguments that the method requires.
 - comma-separated list of parameter descriptors
- Each descriptor is a type and identifier.
 - eg.

```
int count, double len, SomeClass obj, String[] args
```
 - The type is required, just like in variable declarations.

Return Type

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

- The type of info the method returns
 - `void` means no value returned
- Other types require a `return` statement.
 - the value of the result
 - type of return expression must match return type
- eg.

```
public static int double(int num) {  
    return num*2;  
}
```

Modifiers

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

- change the way the function interacts with the rest of the code
- so far, all have been “public static”
 - These become relevant when creating classes...

Example

```
class FunctionTest {  
    public static double negate(double x) {  
        return -x;  
    }  
    public static void main(String[] args) {  
        System.out.println( negate(3.4) + 1 );  
    }  
}
```

- **Output:** (java FunctionTest)
-2.4

Creating Classes

- Syntax:

`<class defn> ::= <modifiers> class <identifier> <associations> { <body> }`

- Example:

```
public class HelloWorld { ... }
```

- The body contains any methods (functions) and data declarations (properties/variables) for the class.
- Typically, the class `HelloWorld` should be in the file `HelloWorld.java`.
 - Will be compiled automatically if used in another class.

Class Libraries

- Classes in Java can be used to hold code libraries
 - ... and also to create objects (later).
- Functions (methods) can be placed in a class file and used in another class.
 - This lets them work like Python modules.

Example 1

■ In Quadratic.java:

```
class Quadratic {
    static private double discrim(double a, double b, double c) {
        return b*b - 4*a*c;
    }
    static public double root1(double a, double b, double c) {
        return (-b - Math.sqrt(discrim(a,b,c))) / (2*a);
    }
    static public double root2(double a, double b, double c) {
        return (-b + Math.sqrt(discrim(a,b,c))) / (2*a);
    }
}
```

■ Note: no main() method

Example 2

■ In QuadTest.java:

```
class QuadTest {  
    public static void main(String[] args) {  
        double a=1, b=-4, c=3;  
        System.out.println( Quadratic.root1(a,b,c) );  
        System.out.println( Quadratic.root2(a,b,c) );  
        // System.out.println( Quadratic.discrim(a,b,c) );  
    }  
}
```

■ Output:

1.0

3.0

Private/Public Members

- Methods and properties with the `public` modifier can be accessed from outside.
 - eg. `root1` and `root2` in `Quadratic`
- `private` methods and properties can't.
 - can only be accessed inside that class.
 - eg. `discrim` in `Quadratic`
 - Calling `discrim` from `QuadTest` would cause a compile error.

Packages

- The Java class library is organized in “packages”.
 - contains a large collection of useful tools
 - See the online reference for details.
- Can be accessed by their full name, or imported
 - see example...
- You can also create your own packages.
 - ...but we won't.

Example

- These are equivalent: (separate classes & files)

```
class DateTest1 {  
    public static void main(String[] args) {  
        java.util.Date d = new java.util.Date();  
        System.out.println(d);  
    }  
}
```

```
import java.util.Date; // or import java.util.* for everything  
class DateTest2 {  
    public static void main(String[] args) {  
        Date d = new Date();  
        System.out.println(d);  
    }  
}
```