CS 241 — Introduction to Problem Solving and Programming

Fundamentals of Programming

Introduction to Methods

Feb 4, 2005

Overview

- Motivation for methods
- Examples and form
- Some terminology and other technical stuff
- Methods as encapsulation and abstraction
- Varying examples

Intro

We have seen some methods already in this class

- `System.out.println(...)`
- `str.charAt(...)`
- `DocsIO.readint(...)`

. . . but we haven't said what methods are.

Repetition

We've seen sequential repetition.

```
public class TempConvert {
    public static void main(String[] args) {
        for (;;) {
            int temp = DocsIO.readint("Please enter an F temp (-1 to quit)--> ");
            if (temp == -1) break;
            double convertedTemp = (5 * ((double) temp -32)) / 9;
            System.out.println("Temperature in C: " + convertedTemp);
        }
    }
}
```

Repetition

But what if we want to repeat some functionality in several, non-sequential places?

```
int temp1 = DocsIO.readint("Please enter the first F temp--> ");
double convertTemp1 = (5 * ((double) temp1 -32)) / 9;
int time = DocsIO.readint("Please enter the elapsed time (in seconds)--> ")
int temp2 = DocsIO.readint("Please enter the second F temp--> ");
double convertTemp2 = (5 * ((double) temp2 -32)) / 9;
System.out.println("Temperature change: "
                   + (convertTemp2 - convertTemp1) / time
                   + " degrees C per second");
```

Repetition

In mathematical terms, we are calculating a function:

$$f(x) \quad = \quad \frac{5 \cdot (x - 32)}{9} \qquad \text{Conversion from F to C}$$

Algorithmically, we want to factor out a piece of functionality, pre-define it, and be able to paste it in.

Algorithmic view

- Predefine **TempConvert:**
  - Subtract 32 from given number.
  - Multiply result by 5.
  - Divide result by 9.
- Read `temp1` from user.
- **TempConvert** `temp1`, store result in `covertTemp1`.
- Read `time` from user.
- Read `temp2` from user.
- **TempConvert** `temp2`, store result in `covertTemp2`.
- Display `(convertTemp2 - convertTemp2) / time`.

Methods

Reusable code can be encapsulated in a method.

A method is a piece of a program that is broken off from the rest of the code because it is self-contained or used repeatedly or because it would make the main part of the program too long or too complicated.

A method is basically the same thing as a function, procedure, or subroutine.

Using a method

```
public class TempConvert {
    public static void main(String[] args) {
        int temp1 = DocsIO.readint("Please enter the first F temp--> ");
        double convertTemp1 = convert(temp1);
        int time = DocsIO.readint("Please enter the elapsed time--> ");
        int temp2 = DocsIO.readint("Please enter the second F temp--> ");
        double convertTemp2 = convert(temp2);
        System.out.println("Temperature change: "
                           + (convertTemp2 - convertTemp1) / time
                           + " degrees C per second");
    }
    static double convert(int temp) {
        return (5 * ((double) temp -32)) / 9;
    }
}
```

Use what's called a method.

Using a method

```
public class TempConvert {
    public static void main(String[] args) {
        int temp1 = DocsIO.readint("Please enter the first F temp--> ");
        double convertTemp1 = convert(temp1);
        int time = DocsIO.readint("Please enter the elapsed time--> ");
        int temp2 = DocsIO.readint("Please enter the second F temp--> ");
        double convertTemp2 = convert(temp2);
        System.out.println("Temperature change: "
                           + (convertTemp2 - convertTemp1) / time
                           + " degrees C per second");
    }
    static double convert(int temp) {
        return (5 * ((double) temp -32)) / 9;
    }
}
```

Places where we call or invoke the method.

Using a method

```
public class TempConvert {
    public static void main(String[] args) {
        int temp1 = DocsIO.readint("Please enter the first F temp--> ");
        double convertTemp1 = convert(temp1);
        int time = DocsIO.readint("Please enter the elapsed time--> ");
        int temp2 = DocsIO.readint("Please enter the second F temp--> ");
        double convertTemp2 = convert(temp2);
        System.out.println("Temperature change: "
                           + (convertTemp2 - convertTemp1) / time
                           + " degrees C per second");
    }
    static double convert(int temp) {
        return (5 * ((double) temp -32)) / 9;
    }
}
```

The method definition—note it is outside the `main` stuff.

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

"Magic" word. . . we'll see what it means later and see some methods that don't need it.

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

The type of the value computed by the method, called the return type.

When a method is called, a value is returned, and we must specify what type should be expected at the place where it is called.

The method call is an expression and therefore has a type.

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

The method's name (an identifier).

The same naming conventions used for variables are used for methods (that is, lowercase, except for internal capitalization when joining words together).

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

A name and type for a passed value, called the parameter or argument. (More on the names later.)

They work like variable declarations. Same naming conventions apply. `temp` is treated like a variable whose scope is the body of the method.

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

The body of the method.

This is a list of statements bound together with curly braces, like a block statement.

The curly braces are not optional.

Using a method

```
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

The keyword `return` marks a return statement.

Return statement: `return` *Expression*;

This specifies what value the method call has.

What happens when a method is called?

```
    double convertTemp1 = convert(temp1);
   . . .
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

The expression is is evaluated. The value is copied into the parameter of the method.

The value of `temp` is set to be the value of `temp1`.

What happens when a method is called?

```
    double convertTemp1 = convert(temp1);
   . . .
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

Some people differentiate between argument and parameter:

Scheme 1	*Scheme 2*	*Meaning*
parameter	formal parameter	The variable used to store the value in the body of the method.
argument	actual parameter	The value passed to the method.

What happens when a method is called?

```
    double convertTemp1 = convert(temp1);
   . . .
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

Control is passed to the body of the method.

Execution proceeds until a return statement is encountered.

What happens when a method is called?

```
    double convertTemp1 = convert(temp1);
   . . .
static double convert(int temp) {
    return (5 * ((double) temp -32)) / 9;
}
```

Control returns to the point where the method is called, with the call evaluating to the returned value.

Methods

Some technical points about methods:

- Methods can be arbitrarily long.
- Methods can have any number of parameters.
- Methods can have multiple return statements (but often it's good style to have only one).
- Methods can contain anything the rest of the program can.
- Methods can call other methods (even, as we'll see next week, themselves).

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);                    x: 1
    System.out.println("y: " + y);                    y: 2
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);                    x: 1
    System.out.println("y: " + y);                    y: 2
    int z = crazy(x, y);                              x: 1
    System.out.println("x: " + x);                    y: 2
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

```
x: 1
y: 2
x: 1
y: 2
x: 2
y: 3
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

```
x: 1
y: 2
x: 1
y: 2
x: 2
y: 3
x: 1
y: 2
z: 5
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

```
x: 1
y: 2
x: 1
y: 2
x: 2
y: 3
x: 1
y: 2
z: 5
x: 2
y: 1
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

```
x: 1
y: 2
x: 1
y: 2
x: 2
y: 3
x: 1
y: 2
z: 5
x: 2
y: 1
x: 3
y: 2
```

Predict a method's behavior

```
public static void main(String[] args) {
    int x = 1;
    int y = 2;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    int z = crazy(x, y);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
    z = crazy(y, x);
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("z: " + z);
}
static int crazy(int x, int y) {
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    x++; y++;
    System.out.println("x: " + x);
    System.out.println("y: " + y);
    return x + y;
}
```

```
x: 1
y: 2
x: 1
y: 2
x: 2
y: 3
x: 1
y: 2
z: 5
x: 2
y: 1
x: 3
y: 2
x: 1
y: 2
z: 5
```

Summary

Be able to identify the following concepts.

- Method
- Call/invocation
- Definition
- Return
- Parameter/argument