Lecture 6: Methods

MIT-AITI Kenya 2005
In this lecture, you will learn…

• What a method is
• Why we use methods
• How to declare a method
• The four parts of a method
• How to invoke a method
• The purpose of the main method
• And see some example methods
The Concept of a Method

• Methods also known as functions or procedures.

• Methods are a way of capturing a sequence of computational steps into a reusable unit.

• Can you think of places where you might want to use methods?
The Concept of a Method

• Methods also known as functions or procedures.

• Methods are a way of capturing a sequence of computational steps into a reusable unit.

• Can you think of places where you might want to use methods?
  – evaluate the quadratic formula, print to the screen
The Concept of a Method (con’t)

• Methods can accept inputs in the form of *arguments*

• They can then perform some operations with the arguments

• And can *return* a value that is the result of the computations, which is also known as the output
Square Root Method

• Square root is a good example of a method.

• The square root method accepts a single number as an argument and returns the square root of that number.
Square Root Method (con’t)

• The computation of square roots involves many intermediate steps between input and output.
• When we use square root, we don’t care about these steps. All we need is to get the correct output.
• Hiding the internal workings of a method from a user but providing the correct answer is known as *abstraction*.
Methods Pop Quiz

• What is the name given to the inputs of a method?
Methods Pop Quiz

• What is the name given to the inputs of a method?
  – Arguments
Methods Pop Quiz

• What is the name given to the inputs of a method?
  – Arguments

• Why do we use methods?
Methods Pop Quiz

• What is the name given to the inputs of a method?
  – Arguments

• Why do we use methods?
  – To capture a sequence of steps which can
    – later be reused
Methods Pop Quiz

• What is the name given to the inputs of a method?
  – Arguments

• Why do we use methods?
  – To capture a sequence of steps which can
    – later be reused

• What is the name given to hiding the internal workings of a method?
Methods Pop Quiz

• What is the name given to the inputs of a method?
  – Arguments

• Why do we use methods?
  – To capture a sequence of steps which can
  – later be reused

• What is the name given to hiding the internal workings of a method?
  – Abstraction
Declaring Methods

• A method has 4 parts: the return type, the name, the arguments, and the body:

```
double sqrt(double num) {
    // a set of operations that compute
    // the square root of a number
}
```

• The type, name and arguments together is referred to as the **signature** of the method
The Return Type of a Method

• The return type of a method may be any data type.

• The type of a method designates the data type of the output it produces.

• Methods can also return nothing in which case they are declared void.
Return Statements

• The return statement is used in a method to output the result of the methods computation.
• It has the form:
  - `return expression_value;`
• The type of the expression_value must be the same as the type of the method:
  ```java
double sqrt(double num) {
    double answer;
    // Compute the square root of num
    // and store in answer
    return answer;
}
```
Return Statements (con’t)

• A method exits immediately after it executes the return statement

• Therefore, the return statement is usually the last statement in a method

• A method may have multiple return statements. Can you think of an example of such a case?
Multiple Returns

• An example using multiple returns:

```c
int absoluteValue (int num) {
    if (num < 0)
        return -num;
    else
        return num;
}
```
void Methods

• A method of type `void` has a return statement without any specified value. i.e. `return;`

• This may seem useless, but in practice void is used often.

• A good example is when a methods only purpose is to print to the screen.

• If no return statement is used in a method of type void, it automatically returns at the end
Method Arguments

• Methods can take input in the form of arguments.

• Arguments are used as variables inside the method body.

• Like variables, arguments must have their type specified.

• Arguments are specified inside the parentheses that follow the name of the method.
Here is an example of a method that divides two doubles:

define divide(double a, double b) {
    double answer;
    answer = a / b;
    return answer;
}
Method Arguments

• Multiple method arguments are separated by commas:
  ```java
double pow(double x, double y)
  ```

• Arguments may be of different types
  ```java
int indexOf(String str, int fromIndex)
  ```
The Method Body

• The body of a method is a block specified by curly brackets i.e \{ \}. The body defines the actions of the method.

• The method arguments can be used anywhere inside of the body.

• All methods must have curly brackets to specify the body even if the body contains only one statement or no statements.
Invoking Methods

• To call a method, specify the name of the method followed by a list of comma separated arguments in parentheses:
  \[\text{pow}(2, 10); \text{//Computes } 2^{10}\]

• If the method has no arguments, you still need to follow the method name with empty parentheses:
  \[\text{size}();\]
Static Methods

• Some methods have the keyword `static` before the return type:

  ```java
  static double divide(double a, double b) {
    return a / b;
  }
  ```

• We'll learn what it means for a method to be static in a later lecture

• For now, all the methods we write in lab will be static.
main – A Special Method

• The only method that we have used in lab up until this point is the `main` method.

• The main method is where a Java program always starts when you run a class file

• The `main` method is static and has a strict signature which must be followed:

```java
public static void main(String[] args) {
    ...
}
```
main Method (con’t)

class SayHi {
    public static void main(String[] args) {
        System.out.println("Hi, " + args[0]);
    }
}

• If you were to type `java Program arg1 arg2 ... argN` on the command line, anything after the name of the class file is automatically entered into the `args` array:

    java SayHi Sonia

• In this example `args[0]` will contain the String "Sonia", and the output of the program will be "Hi, Sonia".
Methods Pop Quiz 2

• What are the four parts of a method and what are their functions?
Methods Pop Quiz 2

• What are the four parts of a method and what are their functions?
  Return type – data type returned by the method
  Name – name of the method
  Arguments – inputs to the method
  Body – sequence of instructions executed by the method
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?

Comma
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?
  Comma

• What is used to outline the body of a method?
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?
  Comma

• What is used to outline the body of a method?
  Curly brackets { }
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?
  Comma

• What is used to outline the body of a method?
  Curly brackets { }

• How do you invoke a method?
Methods Pop Quiz 2 (con’t)

• What is used to separate multiple arguments to a method?
  Comma

• What is used to outline the body of a method?
  Curly brackets { }

• How do you invoke a method?
  Specify the name of the method followed by a list of comma-separated arguments in parentheses, i.e. method_name(arg1, arg2, …, argn)
What is wrong with the following?

```java
static double addSometimes(num1, num2) {
    double sum;
    if (num1 < num2) {
        sum = num1 + num2;
        String completed = "completed";
        return completed;
    }
    
}
What is wrong with the following?

```java
static double addSometimes(num1, num2){
    double sum;
    if (num1 < num2){
        sum = num1 + num2;
        String completed = “completed”;    
        return completed;
    }
}
```

- Types for the arguments `num1` and `num2` are not specified
- String `completed` does not match the correct double return type
- Method `addSometimes` does not always return an answer. This will cause an error in Java because we specified that `addSometimes` would always return a double.
Example main method

```java
class Greetings {
    public static void main(String args[]) {
        String greeting = "";
        for (int i=0; i < args.length; i++) {
            greeting += "Jambo " + args[i] + "! ";
        }
        System.out.println(greeting);
    }
}
```

• After compiling, if you type
  java Greetings Alice Bob Charlie
  prints out "Jambo Alice! Jambo Bob! Jambo Charlie!"
Another Example

class Max {
    public static void main(String args[]) {
        if (args.length == 0) return;

        int max = Integer.parseInt(args[0]);
        for (int i=1; i < args.length; i++) {
            if (Integer.parseInt(args[i]) > max) {
                max = Integer.parseInt(args[i]);
            }
        }
        System.out.println(max);
    }
}

• After compiling, if you type `java Max 3 2 9 2 4` the program will print out 9
Important Points Covered ....

• Methods capture a piece of computation we wish to perform repeatedly into a single abstraction

• Methods in Java have 4 parts: return type, name, arguments, body.

• The return type and arguments may be either primitive data types (i.e. int) or complex data types (i.e. Objects), which we will cover next lecture

• **main** is a special Java method which the java interpreter looks for when you try to run a class file

• **main** has a strict signature that must be followed:
  
  ```java
  public static void main(String args[])
  ```