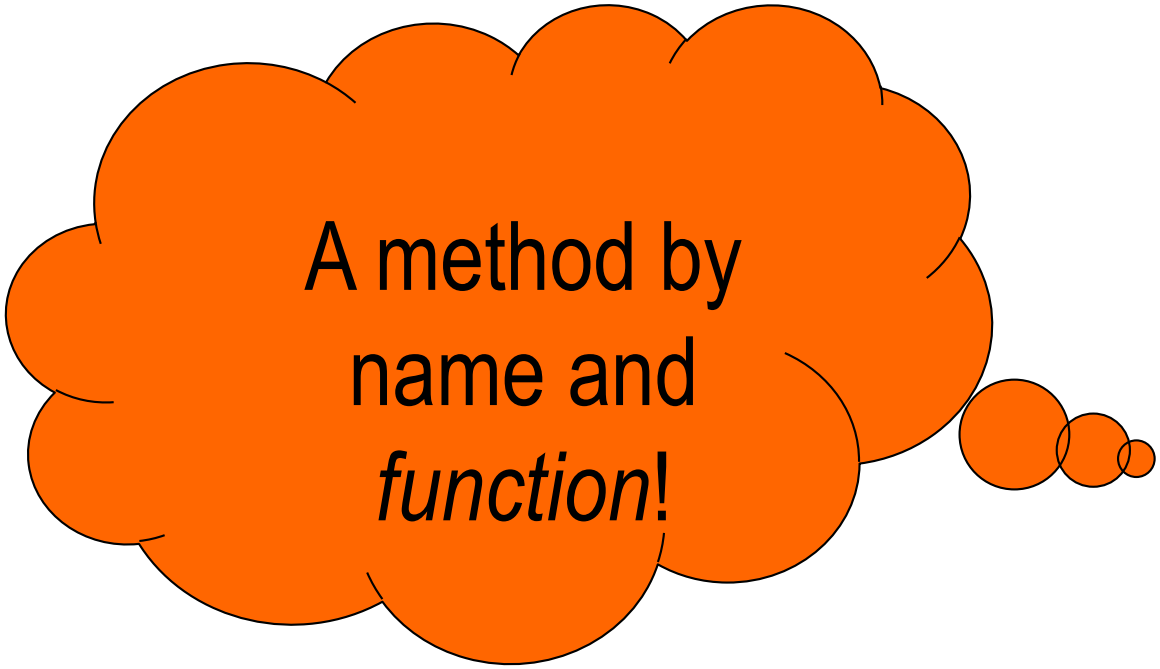


Functions in Java

Computer Science 112
Boston University

Christine Papadakis

Functions / Methods:

Python

- Python distinguishes between:
  - *functions:* named blocks of code that:
    - take 0 or more inputs/parameters, *called arguments*
    - return a value, *explicitly or implicitly*
    - are ***independent*** blocks that can be invoked by calling the name of the function

```
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

Functions / Methods:

Python

- Python distinguishes between:
  - *functions:* named blocks of code that:
    - take 0 or more inputs/parameters, *called arguments*
    - return a value, *explicitly or implicitly*
    - are independent blocks that can be invoked by calling the name of the function
  - ***methods:*** functions that are defined within a class

```
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h

    def area(self):
        a = self.width * self.height
        return a
```

Functions / Methods:

- Python distinguishes
  - *functions:* na
    - take 0 or
    - return a va
    - are independ g the name of the function
  - *methods:* functions that are defi within a class

Methods can only be called on an instance of a class. Example:

```
rect = new Rectangle( 10, 15 )
rect.area()
```

```
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h

    def area(self):
        a = self.width * self.height
        return a
```

Functions / Methods: Python

- Python distinguishes between:
  - *functions:* named blocks of code t
    - take 0 or more inputs/paramet
    - return a value, *explicitly or imp*
    - are independent blocks that can e name of the function
  - *methods:* functions that are defined within ass

The first parameter of each method must be the **self** parameter which references the object the method was called on.

```
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h

    def area(self):
        a = self.width * self.height
        return a
```

Functions / Methods: Python

- Python distinguishes between:
  - *functions:* named blocks of code t
    - take 0 or more inputs/paramet
    - return a value, *explicitly or imp*
    - are independent blocks that can e name of the function
  - *methods:* functions that are "inside" an ect

All attributes must be accessed through the **self** reference!

```
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

```
class Rectang  :
    def __in t__(self, w, h):
        self.width = w
        self.height = h

    def area(self):
        a = self.width * self.height
        return a
```

Functions / Methods

- Python distinguishes between:
  - *functions:* named blocks o
    - take 0 or more inputs,
    - return a value, *explicit*
    - are independent blocks g the name of the function
  - *methods:* functions that are "inside" an object
    - have a `self` parameter

This actually makes sense because everything originates from a class!

- In Java, both types of functions are called methods.

Functions / Methods

- Python distinguishes between:
  - *functions:* named blocks o
    - take 0 or more inputs,
    - return a value, *explicit*
    - are independent blocks g the name of the function
  - *methods:* functions that are "in de" an object
    - have a `self` parameter

But we still need a mechanism to allow us to define functions that act as stand-alone independent entities!

- In Java, both types of functions are called methods.

Functions / Methods

- Python distinguishes between:
  - *functions:* named blocks of code that:
    - take 0 or more inputs/para *uments*
    - return a value, *ex*
    - are independen ng the name of the fu
  - *methods:* function
    - have a `self` pa

More accurately,
closest equivalent to
Python functions!

- In Java, both types of fun ons are called methods.
  - *static* methods ⇔ Python functions

Functions / Methods

- Python distinguishes between:
  - *functions:* named blocks of code that:
    - take 0 or more inputs/parameters, *called arguments*
    - return a value, *explicitly or implicitly*
    - are independent blocks that can be called by using the name of the function
  - *methods:* functions that are "inside" an object
    - have a `self` parameter

- In Java, both types of functions are called methods.
  - *static* methods ⇔ Python functions
  - *non-static* or *instance* methods ⇔ Python methods

Python Functions ⇔ Java Static Methods

Python

```python
def grade(avg):
    if avg >= 90:
        grade = 'A'
    elif avg >= 80:
        grade = 'B'
    elif avg >= 70:
        grade = 'C'
    elif avg >= 60:
        grade = 'D'
    else:
        grade = 'F'

    return grade
```

Java

```java
public static String grade(int avg) {
    String grade;
    if (avg >= 90) {
        grade = "A";
    } else if (avg >= 80) {
        grade = "B";
    } else if (avg >= 70) {
        grade = "C";
    } else if (avg >= 60) {
        grade = "D";
    } else {
        grade = "F";
    }

    return grade;
}
```

- Format of the header of most static methods:

```java
public static return-type method-name(parameters)
```

where each parameter is preceded by its type, { } defines the body

Examples of Static Methods

returns a value of type `double`

```
public static double max(double val1, double val2) {
    if (val1 > val2) {
        return val1;
    } else {
        return val2;
    }
}
```

takes two parameters, both of type `double`

Examples of Static Methods

```
public static double max(double val1, double val2) {
    if (val1 > val2) {
        return val1;
    } else {
        return val2;
    }
}
```

```
val1 = 10.5;
val2 = 20.7;
```

- Here's an example of calling it:

```
double larger = max(10.5, 20.7);
```

Examples of Static Methods

```
public static double max(double val1, double val2) {
    if (val1 > val2) {
        return val1;
    } else {
        return val2;    // return 20.7;
    }
}
```

```
val1 = 10.5;
val2 = 20.7;
```

- Here's an example of calling it:

```
double larger = max(10.5, 20.7);
                    20.7
```

Examples of Static Methods

```
public static double max(double val1, double val2) {
    if (val1 > val2) {
        return val1;
    } else {
        return val2;
    }
}
```

```
val1 = 10.5;
val2 = 20.7;
```

- Here's an example of calling it:

```
double larger = max(10.5, 20.7);        // larger = 20.7;
                    20.7
```

Examples of Static Methods:
what if?

```
public static double max(double val1, double val2) {
    if (val1 > val2) {
        return val1;
    } // ? val1 <= val2

}
```

```
val1 = 10.5;
val2 = 20.7;
```

- Here's an example of calling it:

```
double larger = max(10.5, 20.7);        // larger = 20.7;
                    20.7
```

max method, ***a variation….***

```
public static double max(double val1, double val2) {

    double max_value;      // declare a return variable

    if (val1 > val2) {     // assign the return variable
        max_value = val1;
    } else {
        max_value = val2;
    }

    return( max_value );   // one return statement
}
```

Examples of Static Methods (cont.)

`void` indicates that the method does *not* return a value

```
public static void printSquareInfo(int sideLength, String units) {
    int perim = 4 * sideLength;
    int area = sideLength * sideLength;

    System.out.println("side length = " + sideLength + " " + units);
    System.out.println("perimeter = " + perim + " " + units);
    System.out.println("area = " + area + " " + units + " squared")
}
```

takes an `int` followed by a `String`

Examples of Static Methods (cont.)

```
sideLength = 8;
units = "inches";
```

```
public static void printSquareInfo(int sideLength, String units) {
    int perim = 4 * sideLength;
    int area = sideLength * sideLength;

    System.out.println("side length = " + sideLength + " " + units);
    System.out.println("perimeter = " + perim + " " + units);
    System.out.println("area = " + area + " " + units + " squared")
}
```

- Here's an example of calling it:

```
printSquareInfo(8, "inches");
```

Examples of Static Methods (cont.)

```
sideLength = 8;
units = "inches";
```

```
public static void printSquareInfo(int sideLength, String units) {
    int perim = 4 * sideLength;              // int perim = 32;
    int area = sideLength * sideLength;      // int area = 64;

    System.out.println("side length = " + sideLength + " " + units);
    System.out.println("perimeter = " + perim + " " + units);
    System.out.println("area = " + area + " " + units + " squared")
}
```

- Here's an example of calling it:

```
printSquareInfo(8, "inches");
```

- Here's the output:

```
side length = 8 inches
perimeter = 32 inches
area = 64 inches squared
```

Examples of Static Methods (cont.)

```
sideLength = 8;
units = "inches";
```

```
public static void printSquareInfo(int sideLength, String units) {
    int perim = 4 * sideLength;
    int area = sideLength * sideLength;

    System.out.println("side length = " + sideLength + " " + units);
    System.out.println("perimeter = " + perim + " " + units);
    System.out.println("area = " + area + " " + units + " squared")
}   // reached the end of the method, so return
```

- Here's an example of calling it:

```
printSquareInfo(8, "inches");
```

- Here's the output:

```
side length = 8 inches
perimeter = 32 inches
area = 64 inches squared
```

Examples of Static Methods (cont.)

```
public static void printSquareInfo(int sideLength, String units) {
    int perim = 4 * sideLength;
    int area = sideLength * sideLength;

    System.out.println("side length = " + sideLength + " " + units);
    System.out.println("perimeter = " + perim + " " + units);
    System.out.println("area = " + area + " " + units + " squared")
}
```

- Here's an example of calling it:

```
printSquareInfo(8, "inches");      // no value is returned!
                                   // in Python, would return None
```

- Here's the output:

```
side length = 8 inches
perimeter = 32 inches
area = 64 inches squared
```

Practice: Computing Absolute Value

- Write a method named `abs` for computing the absolute value of a floating-point number `n`:

```
public static double abs(double n) {

    double abs_val = n;

    if (abs_val < 0) {
        abs_val = abs_val * -1;

    return abs_val;
}
```

Calling Java Static Methods

a second look

Calling a *static* method from *another* method **within** the same class?

Calling a *static* method from *another* method **outside** the class?

Calling a static method

```
public class MyMethods {

   public static double max(double val1, double val2) {
      double max_value;

      if (val1 > val2)
         max_value = val1;
      else
         max_value = val2;

      return( max_value )
   } // end of method max

   public static void printSquareInfo(int sideLength, String units){
      int perim = 4 * sideLength;
      int area = sideLength * sideLength;

      System.out.println("side length = " + sideLength + " " + units);
      System.out.println("perimeter = " + perim + " " + units);
      System.out.println("area = " + area + " " + units + " squared")
   } // end of printSquareInfo

} // end of program class
```

How do we call these methods?

Calling a static method

from within the same class

```
public class MyMethods {

   public static double max(double val1, double val2) {
       .
       .
       .
   } // end of method max

   public static void printSquareInfo(int sideLength, String units) {
       .
       .
       .
   } // end of printSquareInfo

   public static void main( String[] ) {

      double maxValue = max( 5.32, 6.32 );
      printSquareInfo( 3, “cm” );

   } // end of main method

} // end of program class
```

Calling Java Static Methods

a second look

Calling a *static* method from *another* method **wthin** the same class?

Calling a *static* method from *another* method **outside** the class?

Calling a static method

from a different class

```
public class TestMyMethods {

    public static void main( String[] ) {

        double maxValue = max( 5.32, 6.32 );
        printSquareInfo( 3, “cm” );

    } // end of main method

} // end of program class
```

These methods
do not exist within the
scope of the program class
TestMyMethods!

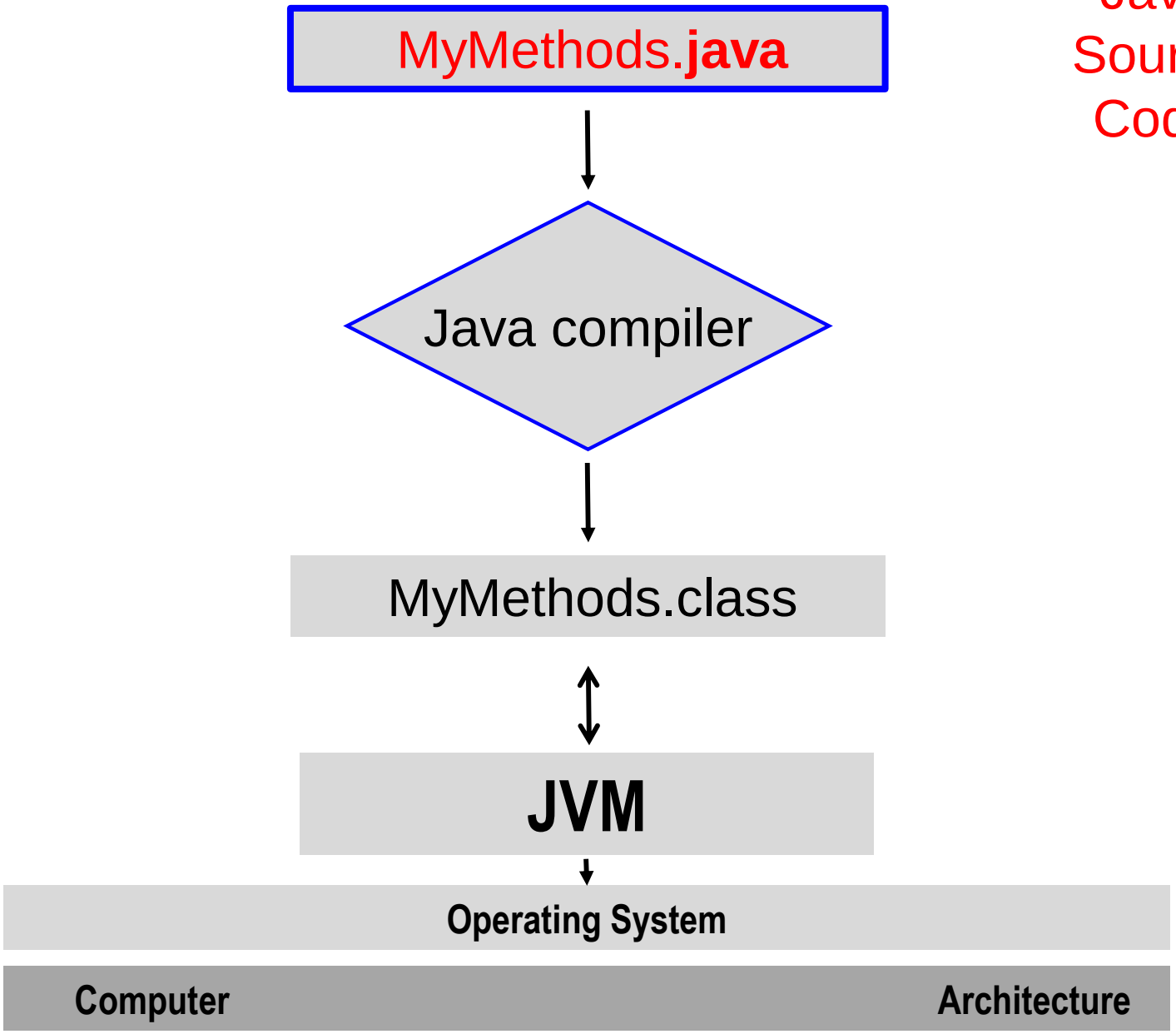
Calling a static method

from a different class

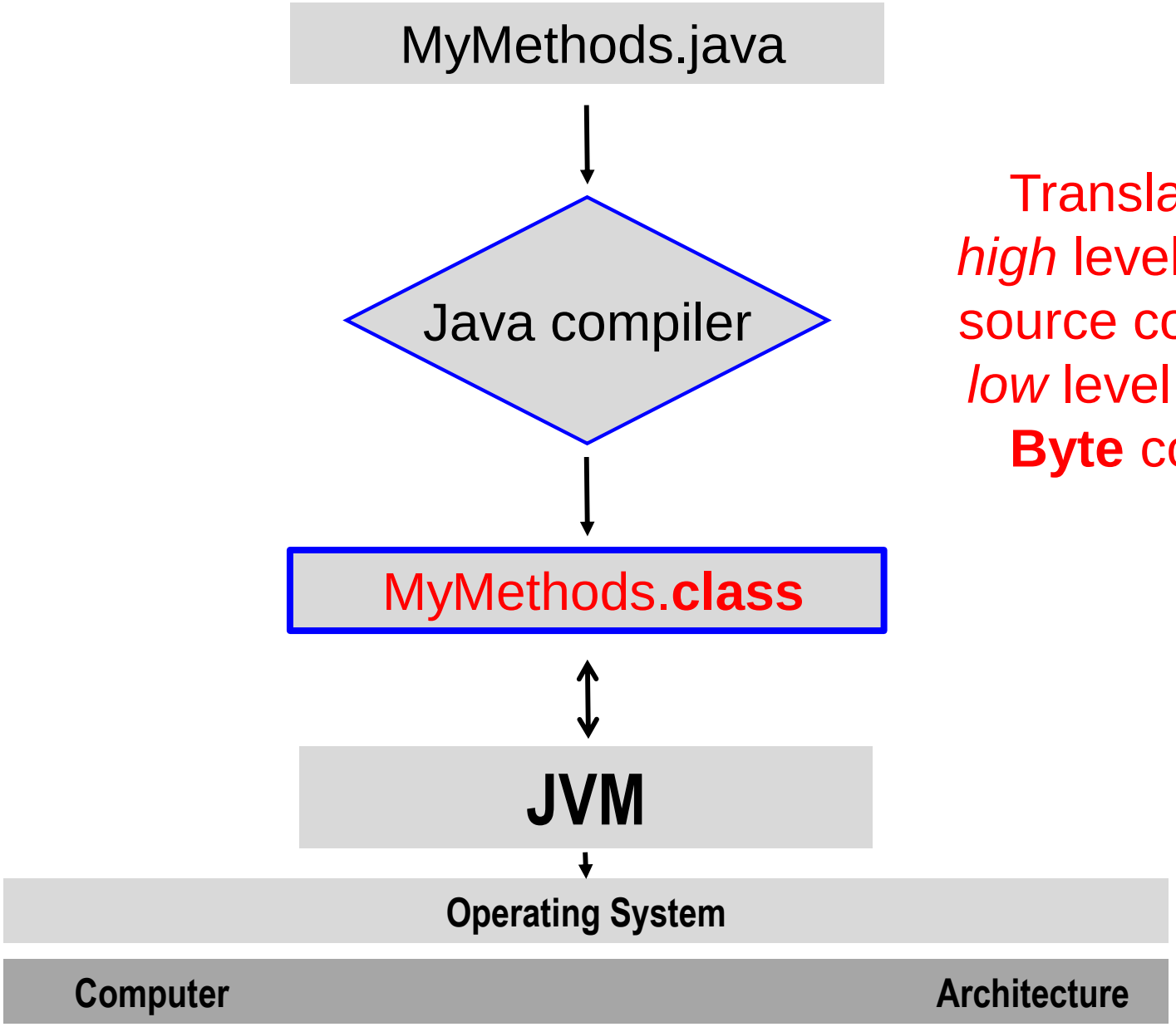
```
public class TestMyMethods {

    public static void main( String[] ) {

        double maxValue = MyMethods.max( 5.32, 6.32 );
        MyMethods.printSquareInfo( 3, “cm” );

    } // end of main method

} // end of program class
```

Have to tell Java which program class they have been defined in.

Executing a Java Program

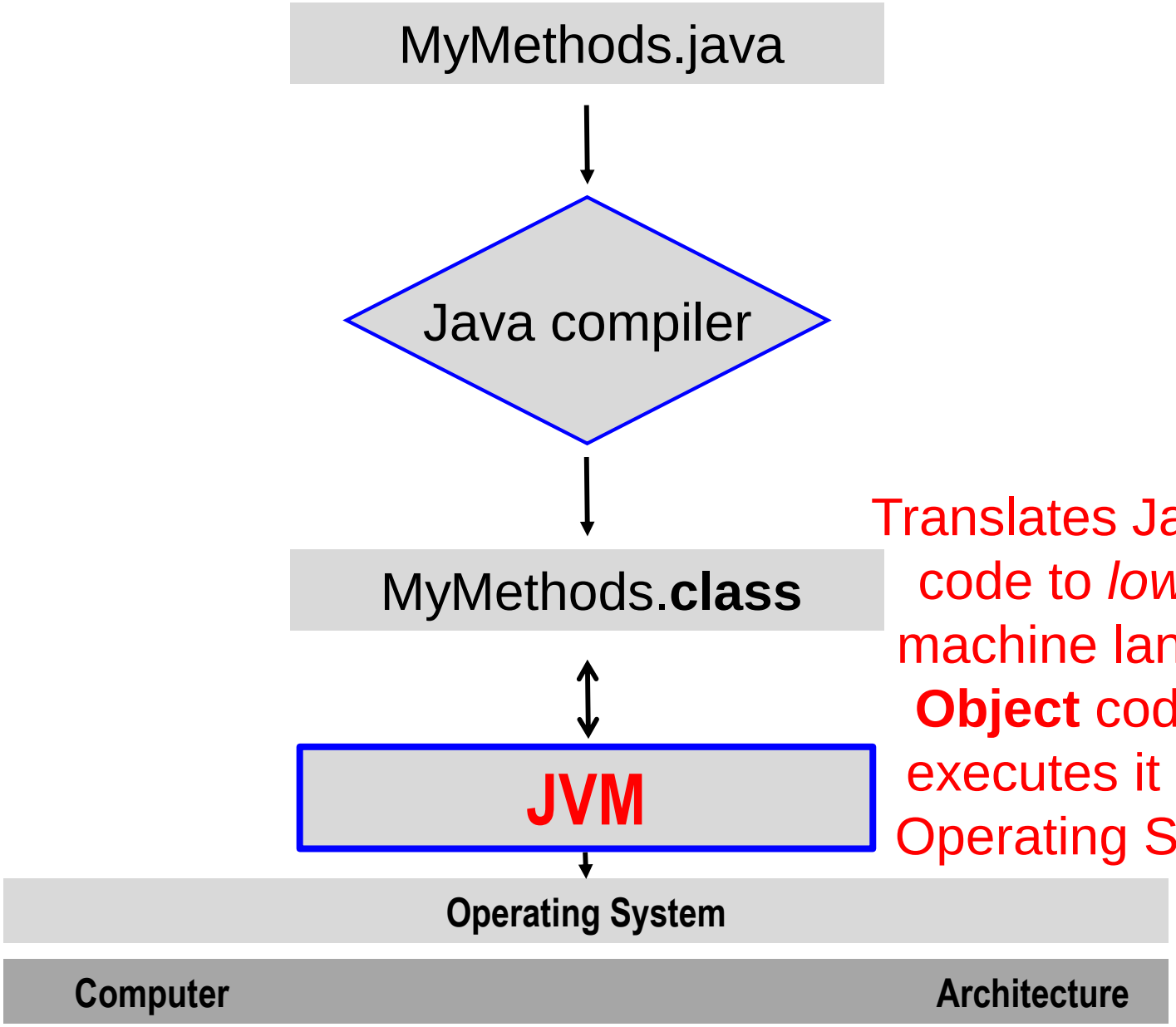


Executing a Java Program



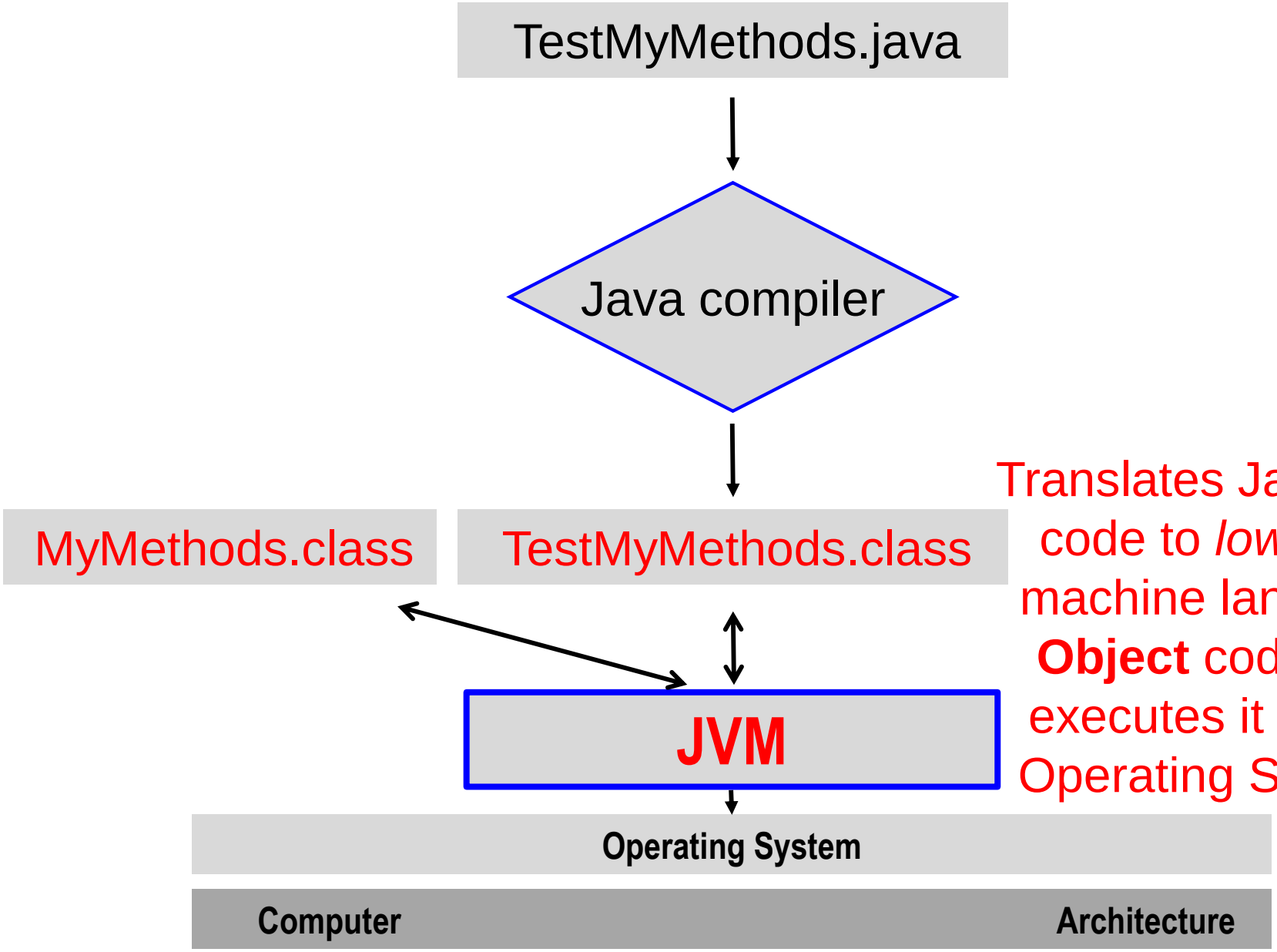
Translates *high* level Java source code to *low* level Java **Byte** code

Executing a Java Program



Translates Java byte code to *low* level machine language **Object** code and executes it on the Operating System.

Executing a Java Program



Translates Java byte code to *low* level machine language **Object** code and executes it on the Operating System.

The `Math` Class

- Java's built-in `Math` class contains static methods for mathematical operations.

- Examples:

  `round(double value)` – returns the result of rounding `value` to the nearest integer

  `abs(double value)` – returns the absolute value of `value`

  `pow(double base, double expon)` – returns the result of raising `base` to the `expon` power

  `sqrt(double value)` – returns the square root of `value`

The `Math` Class (cont.)

- To use a static method defined in another class, we prepend the name of the class.

```
double numVals = Math.pow(2, 20);
```

dot

class name

method name