WIT COMP1000

Method Overloading

Method Overloading

- You can create multiple methods with the same name under certain conditions
- This is called method *overloading*
- Most useful when you need multiple methods that serve a similar purpose but have different parameter requirements
- Note that you can always create methods with different names and avoid overloading altogether, but it might make your program less intuitive to read

Method Overloading

- Multiple methods can have the same name so long as at least one of the following conditions is true
  - The number of parameters is different
  - The types of the parameters are different
- In other words, the parameter list must be different for each overloaded method
  - Only changing the parameter names is not enough!
  - Only changing the return type is not enough!

Example

```
public class ClassExamples {
     public static void main(String[] args) {
          double result1 = max(5.5, 6.7);
          System.out.println("Max of 5.5 and 6.7 is " + result1);
          double result2 = max(7.2, 1.9, -4.5);
          System.out.println("Max of 7.2, 1.9, and -4.5 is " + result2);
     }

     public static double max(double num1, double num2) {
          if (num1 >= num2) {
               return num1;
          }
          return num2;
     }

     public static double max(double num1, double num2, double num3) {
          if (num1 >= num2 && num1 >= num3) {
               return num1;
          } else if (num2 >= num3) {
               return num2;
          }
          return num3;
     }
}
```

Both methods named max( ), but have different number of parameters

Example

```
public class ClassExamples {
    public static void main(String[] args) {
        int remainder = doCalc(42);
        System.out.println("doCalc(42)=" + remainder);
        double fraction = doCalc(42.0);
        System.out.println("doCalc(42.0)=" + fraction);
    }

    public static int doCalc(int x) {
        return x%10;
    }

    public static double doCalc(double x) {
        return x/100;
    }
}
```

Both methods named `doCalc()`, but have parameters of different types

Overloading Rules and Gotchas

- The method used for a particular method call is found by matching the number and type of arguments to the parameter list of each overloaded method
- Methods can't be overloaded when all that is different is the return type
  - » The parameter list must be different!
- An exact match from arguments to parameters is sought out first, then if necessary automatic conversions are used to convert arguments to other types (for example, from `int` to `double`)

Exercise

- Write two methods named `area()`. One should compute the area of a circle and so has a single argument, which is the radius. The second should compute the area of a rectangle and so has two arguments, which are the length and the width. Write a `main()` method to test them both.

Answer

```
public class ClassExamples {
     public static void main(String[] args) {
          double rectangle = area(4.5, 10);
          System.out.printf("Area of rectangle with sides 4.5 and 10 is %.3f%n", rectangle);
          double circle = area(5.2);
          System.out.printf("Area of circle with radius 5.2 is %.3f%n", circle);
     }

     public static double area(double radius) {
          return Math.PI * radius * radius;
     }

     public static double area(double length, double width) {
          return length * width;
     }
}
```

Exercise

- Write three methods named `displayResult()` that print a single value to the screen. One prints an `int` value (as a whole number), one prints a `double` value (with 3 decimal places of accuracy), and one prints a `String` value. Write a `main()` method to test all three.

Answer

```
public class ClassExamples {
    public static void main(String[] args) {
        displayResult(42);
        displayResult("%n");
        displayResult(4.2);
    }

    public static void displayResult(int value) {
        System.out.printf("%d", value);
    }
    public static void displayResult(double value) {
        System.out.printf("%.3f", value);
    }
    public static void displayResult(String value) {
        System.out.printf(value);
    }
}
```

Take Home Points

- Multiple methods can have the same name but different definitions
  - Called method overloading
- Overloaded methods must have different parameter lists
- Mostly used when doing similar calculations with different types of arguments
- Can't overload methods changing only the return type