Objects and Methods

# Constructors

* A **constructor** is a special method that is used to create and initialize an object.
* Using the ***new*** keyword calls a constructor. Ex. **MyClass mc = new MyClass();**
  + MyClass mc declares the variable mc to be a name for an object of the class MyClass.
  + MyClass() creates and initializes a new object, whose address is then assigned to mc.
  + MyClass() is a class to the constructor that Java provided for the class.
  + The parentheses are empty because this constructor takes no arguments.
* A constructor does not have a *return* type.
* A constructor can call methods within its class.
* Sample usages of constructors:
  + public Pet() {

petName = “No name yet”; petAge = 0;

petWeight = 0;

}

//default constructor - has no parameters

* + public Pet(String initialName) {

petName = initialName; petAge = 0;

petWeight = 0;

}

//initializes name of pet only

* + public Pet(int initialAge) {

petName = “No name yet”; petAge = initialAge; petWeight = 0;

}

//initializes age of pet only

* + public Pet(double initialWeight) {

petName = “No name yet”; petAge = 0;

petWeight = initialWeight;

}

//initializes weight of pet only

* + public Pet(String initialName, int initialAge, double initialWeight) { petName = initialName;

petAge = initialAge; petWeight = initialWeight;

}

//initializes name, age, and weight of pet

# Static Variables and Static Methods

* Static variables and methods belong to a class as a whole and not to an individual object.
* A static variable is shared by all the objects of its class.
* A static variable can be public or private.
* Static variables that are not constants should normally be private and should be accessed or changed only by accessor and mutator methods.
* A **static method** is a method that can be invoked without using any object. It is invoked by using the class name instead of an object name.
* A static method is written with the ***static*** keyword in the heading of the method definition.

 When you call a static method, you write the class name instead of the object name

# Ex. inches = UnitConverter.convertFeetToInches(2.6);

*UnitConverter* is the name of the class while *convertFeetToInches()* is the static method.

* A static method cannot reference an instance variable of the class. It cannot invoke a non-static method of the class, unless it has an object of the class and uses the object in the invocation.
* The predefined *Math* class provides a number of standard mathematical methods. The use of *import*

statement is not required.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Description** | **Argument Type** | **Return Type** | **Example** | **Value Returned** |
| pow | Power | double | double | Math.pow(2.0, 3.0) | 8.0 |
| abs | Absolute value | int, long, float, or double | Same as the arg type | Math.abs(-7) Math.abs(7) Math.abs(-3.5) | 7  7  3.5 |
| max | Maximum | int, long, float,  or double | Same as the  arg type | Math.max(5, 6)  Math.max(5.5, 5.3) | 6  5.5 |
| min | Minimum | int, long, float, or double | Same as the arg type | Math.min(5, 6)  Math.min(5.5, 5.3) | 5  5.3 |
| random | Random number | None | double | Math.random() | Random number in the range >0  and <1 |
| round | Rounding | float or  double | int or long respectively | Math.round(6.2) Math.round(6.8) | 6  7 |
| ceil | Ceiling | double | double | Math.ceil (3.2)  Math.ceil(3.9) | 4.0  4.0 |
| floor | Floor | double | double | Math.floor(3.2) Math.floor(3.9) | 3.0  3.0 |
| sqrt | Square root | double | double | Math.sqrt(4.0) | 2.0 |

Example usage: **int higherNum = Math.max(7, 9);**

# Overloading

* **Overloading** occurs when multiple methods have the same name within the same class.
* This is done by having different method definitions in the methods’ parameter lists.
* Examples are:
  + public static double getAverage(double n1, double n2) { }
  + public static double getAverage(double n1, double n2, double n3) { }
  + public static int getAverage(int n1, int n2, int n3) { }
* Java distinguishes methods according to the number of parameters and the types of the parameters.
* A method’s name and the number and types of its parameters are called the method’s **signature**.
* A class cannot define multiple methods with the same signature.
* Constructors can be overloaded too.

# References:

Baesens, B., Backiel, A. & Broucke, S. (2015). *Beginning java programming: The object-oriented approach*.

Indiana: John Wiley & Sons, Inc.

Farrell, J. (2014). *Java programming, 7th Edition*. Boston: Course Technology, Cengage Learning

Savitch, W. (2014). *Java: An introduction to problem solving and programming, 7th Edition*. California: Pearson Education, Inc.