

# Cheat Sheet

## Java Coding Standards

- Always include class documentation
- **Class** names start with **upper-case** letters
- **Variable** names start with **lower-case** letters
- Align { and matching } in same column
- Indent methods and statements
- Use meaningful variable names
- Use **camelNotation**

## Program Structure

Every Java program is a **class**

Program execution begins with the **main method** (if it exists)

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
        System.out.println("Go PARTs!");  
    }  
}
```

- `class` tells Java that this is a class definition
- `public` who can access the class
- `HelloWorld` the name of the class and the file
- `{ }` defines the start and end of the class code

```
public static void main(String[] args) {  
    System.out.println("Hello World!");  
    System.out.println("Go PARTs!");  
}
```

- `main` name of the method
- `public` who can access the method
- `static` type of method
- `void` the method doesn't return a result
- `{ }` defines the start and end of the method code
- `(String[] args)` parameter (data passed to the method)

```
System.out.println("Hello World!");
```

- `System.out.println` built-in method to display data
- `"Hello World!"` argument passed to `println` method
- `;` tells Java where the statement ends

```
public class Foobar {
    public static void main(String[] args) {
        int totalScore = 0;
        System.out.println("Score" + totalScore);
    }
}
```

- You must declare all variables
- This means telling Java the:
  - type: `int`
  - identifier name: `totalScore`
  - initial value (optional): `0`
- By convention variables start with lowercase letters in Java
- This is called camelCase

# Objects

- An **object** is something about which you have data
- Could be a Person:
  - Student
  - Employee
  - a Place:
    - City
    - State
  - or a Thing:
    - Screen Display
    - Game Character
    - Bank Account

# Classes

```
public class Hello {  
    String myString = "";  
    public Hello () {  
        this.myString = "Default string";  
    }  
    public Hello (String myString) {  
        this.myString = myString;  
    }  
    public String getMyString () {  
        return myString;  
    }  
    public void printMyString () {  
        System.out.println(myString);  
    }  
}
```

- A **class** defines the behaviors associated with an object by means of methods
- A **class** stores the state of an object by means of member variables
- A **class** consist of code...
  - statements
    - System.out.println();
  - member variables
    - int theAnswer;
  - zero, one, or more constructors...
    - constructors create objects
  - and zero, one, or more methods
    - methods modify or return object data

## Member Variables

```
public class River {  
    private String name;  
    private int length;  
    private int locks;  
    ...  
}
```

- Member variables contain data associated with an object
- Specify private access to member variables
  - prevents class users from randomly modifying member variable values

# Constructor

```
public class River {  
    ...  
    public River(String inName, int inLen) {  
        name = inName;  
        length = inLen;  
    }  
    ...  
}
```

- Special method that creates an instance of a class, this runs when the class is instantiated
  - **Instantiated** - represent as or by an instance.
- Must have **same name** as the **class**
- May include parameters

# Instantiating an Object

```
public class RiverTest {  
    public static void main(String[] args) {  
        River riverA;  
        riverA = new River("Ohio",981);  
        River riverB = new River("Kanawha",3000);  
    }  
}
```

`riverA` contains hash-code identifier of a particular `River` object, located somewhere in memory

# Accessor Methods

```
public class River {  
    private String name;
```

```
private int length;

...

public String getName() {
    return name;
}

}
```

- retrieves value of object member variable
- usually doesn't accept parameters
- naming convention:
  - `getFooBar` - return a value
  - `findFooBar` - return a calculated result

## Mutator Methods

```
public class River {
    private String name;
    private int length;

    ...

    public void setName(String name) {
        this.name = name;
    }
}
```

- changes value of object member variable
- usually doesn't return a value
- naming convention:
  - `setFooBar`

## Arrays

- Arrays are a sequence of values of the same datatype or class
- Use standard variable declaration except include `[]`
- `int[] grades;`
- Instantiate the array with the desired size using the new operator:

```
grades = new int[25];
String[] myNames = new String[42];
```

- Array `index` values start at 0

- `length` contains the size of an array:
  - `System.out.println("Size: " + myNames.length);`
  - Multi-dimensional arrays specified by repeated `[ ]`
    - `int[][] matrix = new int[25][42];`
- 

Revision #1

Created 24 April 2025 21:24:32 by Brandon Duke

Updated 24 April 2025 21:24:46 by Brandon Duke