Java Boot Camp

‘Just Enough Java to be Dangerous’

Mr. Michaud

Marist School

Java Topics

- Data Types
- Data Structures
- Operators
- Class
- Fields
  - Public
  - Private
- Constructor
- Methods
- Return Type Functions
- Control Structures

In Java – we use Classes to create 'Blueprints' of objects we use in programs.

Three Types of Programming:

A. Linear – Algorithmic
B. Methoding
C. Modeling

A Side Note: Comments

- Comments are for humans
- Computer ignores comments
- Use lots of them in programming
  - Graded on Comments
  - Good style
  - Allows code to be used by others
- // Single Line Comment

- /* Multi line comments
- */

Data Types: Numbers

- **byte:** 8 bit whole number with value from -128 to 127
- **short:** 16 bit whole number from -32,768 to 32, 767
- **int:** 32 bit “integer” whole number ranging from -2,147,483,648 to 2,147,483,647
- **long:** 64 bit number
- **double:** 64 bit floating point number
- **float:** 32 Bit floating point number

From: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Data Types: Other

- **boolean**: 1 Bit value storing ‘true’ or ‘false’ (Sometimes ‘1’ or ‘0’)
- **char:** 16 Bit Unicode character ranging from \u0000 to \uffff (Hexidecimal)
  - This is a single Letter or character
- **String:** (This is actually a class)
  - java.lang.String
  - ‘string’ of characters: like “Mr. Michaud”

Declaring Data

- Example:

```
int myAge = 14;
```

Type

Name of Variable

Value assigned to variable

Other Examples:

```
String greetings = “Hello Apps Class”;
char letter = “a”;
boolean state = false;
```

Data Structures: Array

- Array
  - Fixed Length
  - Holds multiple values of Type of Object
  - Example:

```
String names [] = {“Rebecca”, “Joshua”, “Sandy”};
```

- Index Variable: Refers to the location of a value within the Array

```
names[0] = “Rebecca”;
names[1] = “Joshua”;
names[2] = “Sandy”;
```

- “.len” will return the length or number of elements in an Array

```
names.len   // will return 3
```

Data Structures: ArrayList

ArrayLists are mutable

```
List<String> names = new ArrayList<String>;

names.add(“Joshua”);
names.add(“Rebecca”);
names.add(“Sandy”);

names.size();  // will return 3
names.get(0);  // will return “Joshua”
names.get(1);  // will return “Rebecca”
names.get(2);  // will return “Sandy”
```

Operators

- `String name = “Bob”;`

// Used to Assign a value

- `name == “Bob”` // ‘is equal to’
- `4 * 5` // multiply
- `20 / 4` // Divide
- `4 + 5` // Addition
- `4 – 5` // Subtraction
- `5 % 4` // Modulo
- `5 > 4`
- `4 < 5`
- `5 >= 4`
- `4 <= 5`

Logic

- && // Means “and”
- || // Means “or”

Class

- Blueprint for an object in a Java Program
- Represent a real world object or a piece within your program
- Components:
  - Fields
  - Constructor
  - Methods
    - Accessors ('Getters')
    - Modifiers ("Setters')

Example: Player

```
1  // Sample Class
2
3  public class Player {
4      // Fields
5      private int position;
6      private String name;
7
8      // Constructor
9      public Player(int p, String n) {
10         position = p;
11         name = n;
12     } // end Constructor
13
14 } // End Class Player
15
```

Class, Fields, and Constructor

```
1  // Sample Class
2
3  public class Player {
4      // Fields
5      private int position;
6      private String name;
7
8      // Constructor
9      public Player(int p, String n) {
10         position = p;
11         name = n;
12     } // end Constructor
13
```

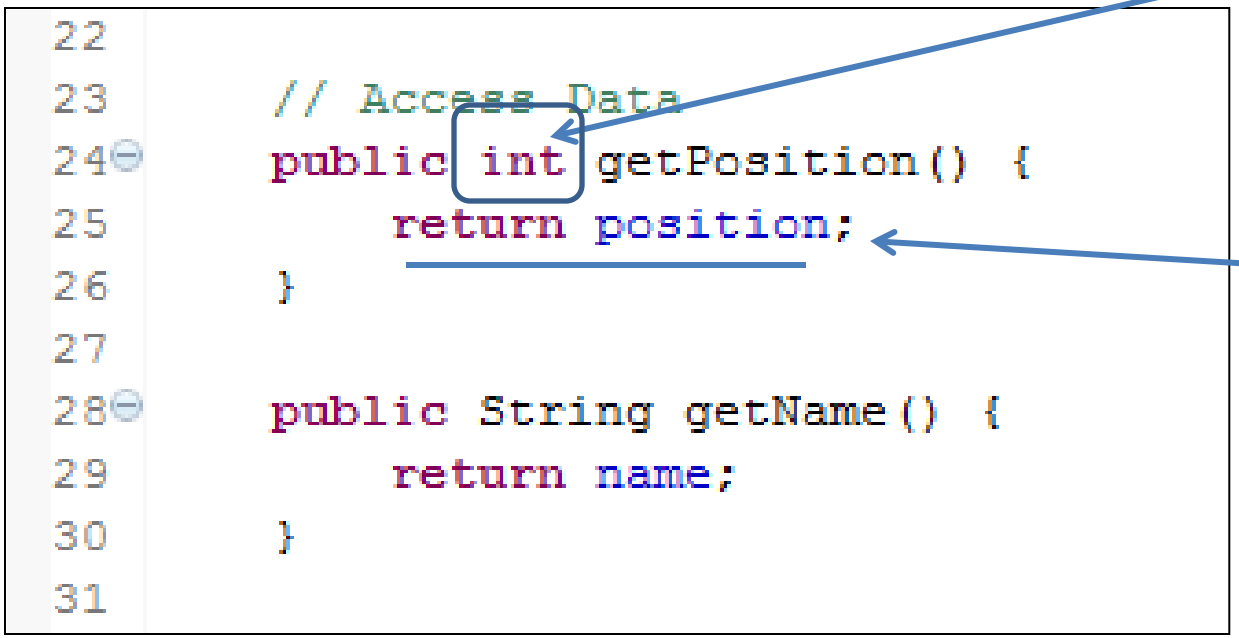
Methods: Modifiers, Access Data, and Method

```
13
14      // Modifiers
15      public void setPosition(int p) {
16          position = p;
17      }
18
19      public void setName(String n) {
20          name = n;
21      }
22
23      // Access Data
24      public int getPosition() {
25          return position;
26      }
27
28      public String getName() {
29          return name;
30      }
31
32      // Move
33      public void move(int m) {
34          position = position + m;
35      }
36
37  } // End Class Player
38
39
```

A Method (or Function) performs a series of commands or operations within a class.

A Class's Methods are the 'actions' or 'procedures' the class knows how to do.

Return Type Functions

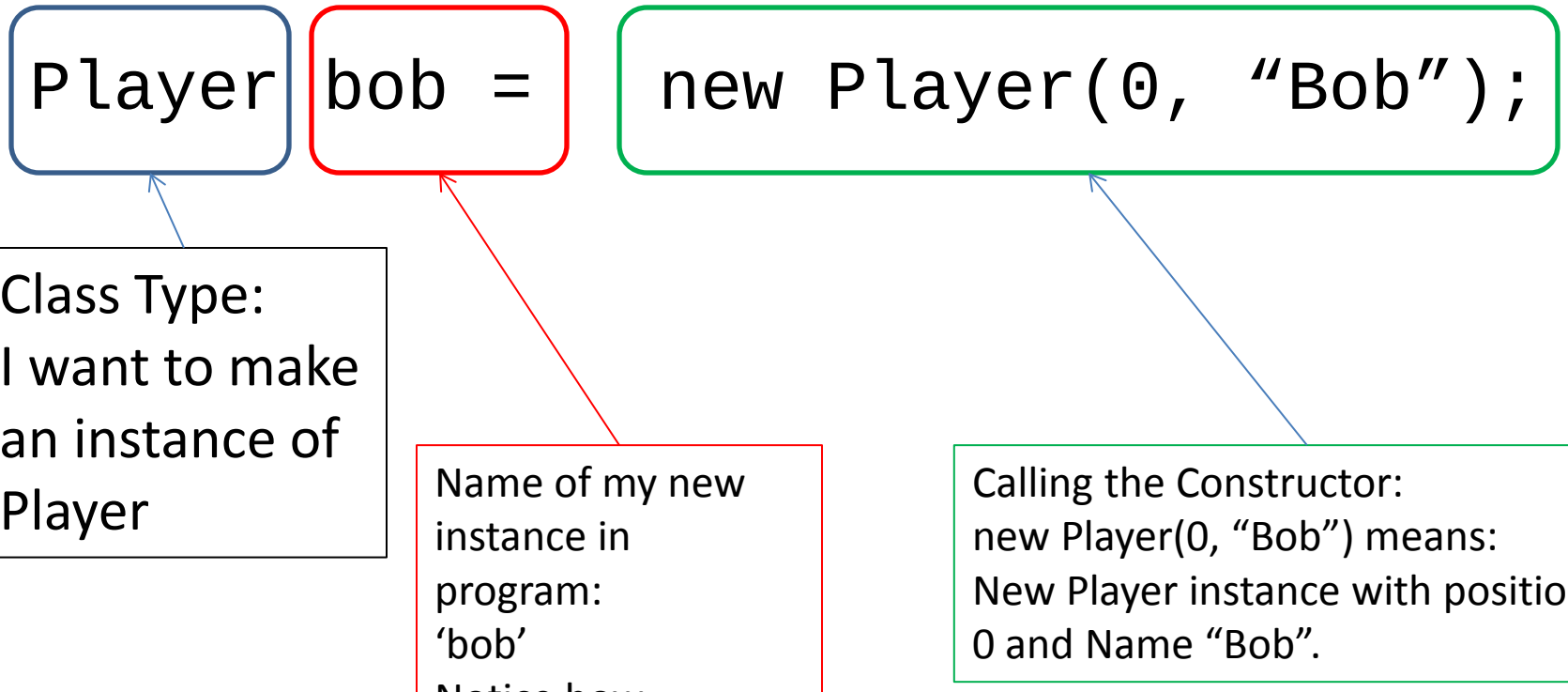


Function return type defined before name:

Return Functions always have a return statement.

Return Type Functions “give back” a value as determined by the data type referenced before the name. An ‘int’ function will return an integer. A ‘String’ function will return a string of characters.

Creating an instance of a Class



Creating an instance and 'calling' the Methods

```
1  // Display Data
2  public class Display {
3
4      public static void main(String[] args) {
5          // Create new Player instance named bob
6          Player bob = new Player(0, "Bob");
7          System.out.println(bob.getName());
8          System.out.println(bob.getPosition());
9          // Move
10         bob.move(4);
11         System.out.println(bob.getPosition());
12     } // end main
13 } // end Class Display
14
15
```

For Loop

- Repeats section of code while counting up or down with an index variable
- Example

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

Returns:
0
1
2
3
4
5
6
7
8
9

for (int i = 0; i < 10; i++) {}

- i++ means "i = i + 1"
- int i means "integer i"

- for (int i = 0; i < 10; i++) means "For index variable i starting at 0, while i is less than 10, count be 1."

Conditional Statements

- ‘if statement’: Checks if a given statement or expression is true and then executes a section of code

```
if (score > 9) {
      textScore.setText(“You Win”);
}
```

While Loop

- Executes a Segment of Code while a Condition is True

```
// Initialize integer score as 0
int score = 0;

// Loop while score is less than 20
while (score < 20) {
    // Print the Score
    System.out.println("Your Score is: " + String.valueOf(score));
    score++; // Increase Score by 1
} // end while

System.out.println("All Done!");
```