JAVA REPORT

Date:	**10/06/2020**	**Name:**	**SAFIYA BANU**
Course:	**JAVA**	**USN:**	**4AL16EC061**
Topic:	1. Arrays of Strings 2. Multi-Dimensional Arrays 3. Classes and Objects 4. Methods 5. Getters and Return Values 6. Method Parameters 7. Setters and "this" 8. Constructors 9. Static (and Final) 10. String Builder and String Formatting	**Semester & Section:**	**8TH B**
Github Repository:	**Safiya-Courses**		

AFTER NOON SESSION DETAILS
ARRAY OF STRINGS Java String array is used to hold fixed number of Strings. String array is very common in simple java programs, specially among beginners to java and to test some specific scenarios. Even java main method argument is string array – **public static void main(String[] args)**
Multidimensional Arrays A multidimensional array is an array containing one or more arrays. To create a two-dimensional array, add each array within its own set of **curly braces**:

Example

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

Classes/Objects

Java is an object-oriented programming language.

Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword `class`:

MyClass.java

Create a class named "`MyClass`" with a variable x:

```
public class MyClass {   int x =

5;


}
```

Create an Object

In Java, an object is created from a class. We have already created the class named `MyClass`, so now we can use this to create objects.

Example

Create an object called "myObj" and print the value of x:

```
public class MyClass {   int x = 5;   public static void

main(String[] args) {


   MyClass myObj = new MyClass();

   System.out.println(myObj.x);

 }

}
```

Java Methods

A **method** is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as **functions**.

Why use methods? To reuse code: define the code once, and use it many times.

Create a Method

A method must be declared within a class. It is defined with the name of the method, followed by parentheses **()**. Java provides some pre-defined methods, such as System.out.println(), but you can also create your own methods to perform certain actions:

Example

Create a method inside MyClass:

```
public class MyClass {   static void

myMethod() {     // code to be executed


  }

}
```

Parameters and Arguments

Information can be passed to methods as parameter. Parameters act as variables inside the method.

Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a String called **fname** as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:

Example

```
public class MyClass {   static void myMethod(String

fname) {


   System.out.println(fname + " Refsnes");

  }    public static void main(String[] args) {

myMethod("Liam");     myMethod("Jenny");

myMethod("Anja");


}

}
```

THIS OPERATOR

Definition and Usage

The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter). If you omit the keyword in the example above, the output would be "0" instead of "5".

this can also be used to:

- Invoke current class constructor
- Invoke current class method
- Return the current class object
- Pass an argument in the method call
- Pass an argument in the constructor call

GETTERS AND SETTERS

private variables can only be accessed within the same class (an outside class has no access to it). However, it is possible to access them if we provide public **get** and **set** methods.

The get method returns the variable value, and the set method sets the value.

Syntax for both is that they start with either get or set, followed by the name of the variable, with the first letter in upper case:

Example

```
public class Person {

 private String name; // private = restricted access



 // Getter

 public String getName() {

  return name;

 }
```

```
  // Setter

  public void setName(String newName) {

    this.name = newName;

  }

}
```

The `get` method returns the value of the variable `name`.

The `set` method takes a parameter (`newName`) and assigns it to the `name` variable. The `this` keyword is used to refer to the current object.

However, as the `name` variable is declared as `private`, we **cannot** access it from outside this class:

Example

```
public class MyClass {

  public static void main(String[] args) {

    Person myObj = new Person();

    myObj.name = "John";  // error

    System.out.println(myObj.name); // error

  }

}
```

Java return Keyword

Example

A method with a return value:

```
public class MyClass {

  static int myMethod(int x) {

    return 5 + x;
```

```
  }

  public static void main(String[] args) {
    System.out.println(myMethod(3));
  }
}
// Outputs 8 (5 + 3)
```

Definition and Usage

The `return` keyword finished the execution of a method, and can be used to return a value from a method.

Java Constructors

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

Example

Create a constructor:

Result Size: 498 x 443

MyClass.java

```java
// Create a MyClass class
public class MyClass {
  int x;

  // Create a class constructor for the MyClass class
  public MyClass() {
    x = 5;
  }

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

Result:

```
5
```

STATIC

Definition and Usage

The static keyword is a non-access modifier used for methods and attributes. Static methods/attributes can be accessed without creating an object of a class.

static method means that it can be accessed without creating an object of the class, unlike public

String Formatting and String builder

The most common way of formatting a string in java is using String.format(). If there were a “java sprintf” then this would be it.

```
String output = String.format("%s = %d", "joe", 35);
```

For formatted console output, you can use printf() or the format() method of System.out and System.err PrintStreams.

```
System.out.printf("My name is: %s%n", "joe");
```

Create a Formatter and link it to a StringBuilder. Output formatted using the format() method will be appended to the StringBuilder.

```
StringBuilder sbuf = new StringBuilder();

Formatter fmt = new Formatter(sbuf);

fmt.format("PI = %f%n", Math.PI);

System.out.print(sbuf.toString());

// you can continue to append data to sbuf here.
```