Topic: Invocation and use of packages, use of import directive.
OOP concepts involved: Classes, Java Packages, Java CLASSPATH, Java PATH, Class Modifiers, Static Methods.
Programming generic concepts involved: Functions, Data Types, Variables, Access Modifiers.

➢ Theoric introduction

JAVA PACKAGE

A package is simply a container that groups related types (Java classes, interfaces, enumerations and annotations). For example, in core Java, the ResultSet interface belongs to java.sql package. The package contains all the related types that are needed for SQL query and database connection.

Packages help to reserve the class namespace and create maintainable code. The rule of thumb in Java programming is that, only one unique class name is allowed in a Java project.

There are two different types of packages in Java:

- *Built-in Packages*

Built-in packages are existing java packages that come along with the JDK. For example, java.lang, java.util, java.io etc.

- *User-defined Packages*

Java also allows you to create packages as per your need. These packages are called user-defined packages.

HOW TO IMPORT PACKAGES IN JAVA?

Java has *import* statement that allows you to import an entire package (as in earlier examples), or use only certain classes and interfaces defined in the package.

The general form of *import* statement is:

```
import package.name.ClassName;   // To import a certain class only
import package.name.*   // To import the whole package
```

For example,

```
import java.util.Date; // imports only Date class
import java.io.*;      // imports everything inside java.io package
```

The import statement is optional in Java. If you want to use class/interface from a certain package, you can also use its *fully qualified name*, which includes its full package hierarchy.

Here is an example to import package using import statement.

```
import java.util.Date;

class MyClass implements Date {
    // body
}
```

The same task can be done using fully qualified name as follows:

```
class MyClass implements java.util.Date {
    //body
}
```

Note: In Java, *import* statement is written directly after the package statement (if it exists) and before the class definition.

For example,

```
package package.name;
import package.ClassName; // only import a Class

class MyClass {
    // body
}
```

- **SETTING CLASSPATH WITH THE THIRD WAY (***CLASSPATH JAVA COMMAND)*

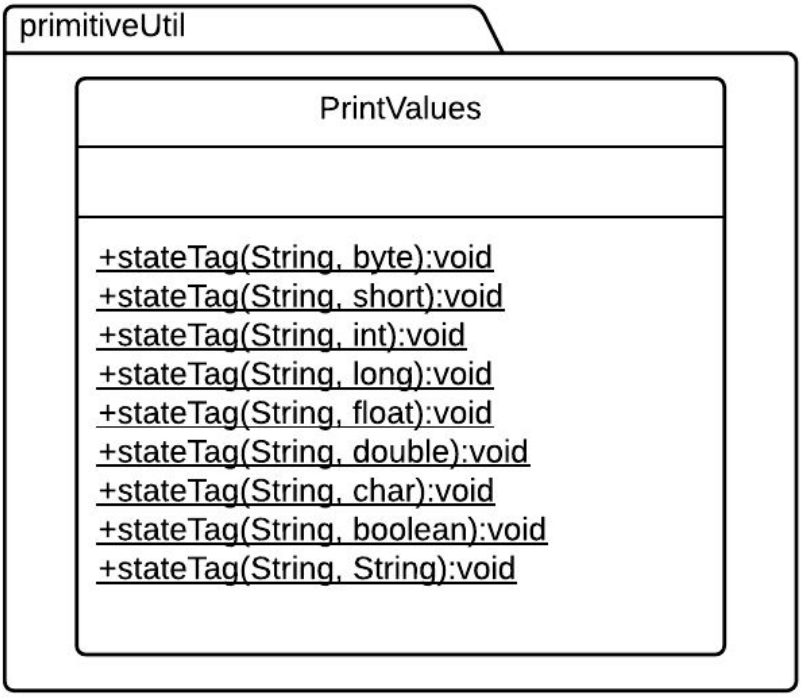
```
C:\javaproject\classes>java -cp C:\javaproject\classes com.figure.Circle
```

```
This is a circle!
```

➢ **Statement**

Develop a package that contains some utility methods for all primitive data types. Start with a method that prints every data type along with a tag. Implement that package into a class and make use of those methods.

➢ **Class design (UML)**



➢ **Program Code**

PrintValues.java

```
package primitiveUtil;

public class PrintValues{

	public static void stateTag(String tag, byte value){
		System.out.println(tag + value);
	}

	public static void stateTag(String tag, short value){
		System.out.println(tag + value);
	}
```

```
	public static void stateTag(String tag, int value){
		System.out.println(tag + value);
	}

	public static void stateTag(String tag, long value){
		System.out.println(tag + value);
	}

	public static void stateTag(String tag, float value){
		System.out.println(tag + value);
	}

	public static void stateTag(String tag, double value){
		System.out.println(tag + value);
	}


	public static void stateTag(String tag, String value){
		System.out.println(tag + value);
	}


	public static void stateTag(String tag, char value){
		System.out.println(tag + value);
	}

	public static void stateTag(String tag, boolean value){
		System.out.println(tag + value);
	}
}
```

Main.java

```
import primitiveUtil.PrintValues;

import java.util.Scanner;
import java.lang.Integer;

public class Main {

	public static void main(String[] args){
		PrintValues.stateTag("String: ", "test");
		PrintValues.stateTag("int: ", 123);
		PrintValues.stateTag("char: ", 'c');
		PrintValues.stateTag("boolean: ", true);
		PrintValues.stateTag("float: ", 2.565f);
	}
}
```

➢ Program execution

During the program execution, we test the package by using with the static stateTag() method with a variety of primitive data types to ensure it works correctly. Since the overloaded *stateTag()* methods are ***static***, it is not necessary to create a ***PrintValues*** object to use them.

```
String: test
int: 123
char: c
boolean: true
float: 2.565
```

➢ Conclusions

Using the Java ***import*** directive to add to our files the packages we will use to access their classes, interfaces, etc. it is a good practice, since it saves us the effort of using the Fully Qualified Package name every time we have to make any reference to the classes associated with the determined package.

The ***import*** directive serves as a short name to use our classes, interfaces, annotations, enumerations, … without many repeated long words (all the package hierarchy).