Java Boot Camp

Manual for Basic Java

By

Document Control

Contacts

Name	Company	Email	Remark
Sunket Ingale	For AgileTestingAlliance.org		

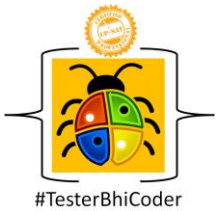
Change Control

Version	Date	Author(s)	Comments
1.0	18 November 2017	SI	Initial version

Table of Content

1 ABOUT JAVA BOOT CAMP AND THIS MANUAL

Java Boot Camp is a part of the larger initiative by Agile Testing Alliance known as **#TesterBhiCoder**.



Agile Testing Alliance has been in favor of building a testing community which is more aware about agile and testing. Today's testing world is changing and the demand for technical testers is far more than anyone else.

ATA wants that all the professionals associated with testing and QA step up and learn coding, be more technical than what they are and start pursuing the next generation role, hence the name **#TesterBhiCoder**

ATA is glad that it is able to help fulfil this objective to a large extent. Most of the folks who have registered for the program are into testing for some time and are eager to move to automation specially selenium.

This learning manual is intended for all the attendees of the Java Boot Camp. In an online session this manual will hopefully help them win over Java.

2 JAVA INTRODUCTION

In the Java programming language, all source code is first written in plain text files ending with the .java extension.

Those source files are then compiled into .class files by the javaccompiler.

A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine (Java VM).

The java launcher tool then runs your application with an instance of the Java Virtual Machine.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on multiple Operating Systems like Windows, Mac, Linux, etc.

3 SET UP AND CONFIGURATION

Download jdk and set the java environment variable

a. Right-click My Computer->Properties. In the System window that opens, click Advance System settings

b. In the Advance System Settings window, click on Environment variables

c. Under SystemVariables, click New and a New System Variable window will be open

d. Under Variable name: specify JAVA_HOME and under Variable value specify the Java path where JDK is installed. In this case C:\Program Files\Java\jdk<version> and press OK button

e. After this scroll in the System Variables, and select the Path variable.

f. Double click the Path Variable and in the Variable Value window at the end type a semicolon and then type %JAVA_HOME%\bin;

g. Verify whether Java is installed successfully by typing the command java –version in the command prompt. This will show the version of java and also means that java is successfully configured

- Download Eclipse from the following link
> http://www.eclipse.org/downloads/eclipse-packages/?osType=win32

- From the downloaded package open eclipse from the eclipse.exe file

- Set the workspace to a folder where all the projects would be saved

4 FIRST PROGRAM

- Create new java project from File > New > Project > Java Project and give your project name. (Note: Make sure java is set to the same version as your jdk version in the execution environment dropdown)

- Click on Finish button

- Right click on project name and select New > Package. Click on finish button.

- Create New Class - Right click on package and create new class with the desired name.

- First Program – 'Hello World'. Type the following:

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

- Run this program by right-click on the class and select Run As Java Application

- The console will print "Hello World"

5 SOURCE CODE COMMENTS

- Comments are ignored by the compiler but are useful to other programmers

- Three ways to comment code

> 1
/ text */*

> 2
/*
The class implements an application that
simply prints "Hello World!" to standard output.
**/*

> 3
// text

6 THE MAIN METHOD

- public static void main(String[] args)

- public – To make the main method accessible to the Compiler and JVM

- static – To call methods in Java we need to make objects of the class. Since this is the main method and it needs to be called by default, we need to define the static keyword

- void - the 'void' keyword is used in Java when the method does not aim to return a value or hold any value.

- main - It's just the name of method. This name is fixed and it's called by the JVM as entry point for an application

- String[] args – Main method can only accept String arguments

7 DATA TYPES

- Data types means which type of value will be stored. For e.g. character, number, decimal etc.

- int a =1234 - Stores smaller numbers but not decimals

- long b = 123456 - Stores larger numbers but not decimals

- double c = 1233.45 - Stores decimal places in variable

- char e = ‘a’ - Stores single character

- boolean g = true - Stores only boolean values like true or false

> Can use AND (&&) , OR(||), NOT(!) operators for Boolean variables

Eg:
1. true && false will return false
2. true || false will return true
3. !true will return false

- String str = “Hello World” - Not a datatype. Useful to store string in variable

8 STRING CLASS

String is not a datatype. It's a built-in class in Java. It has lot of built-in functions to perform in Java.

- str1.equals(str2) - Returns true if both the strings values are same and returns false if both the string values are not same

- str1.concat(str2) - Concatenates str2 with str1. Can also use the '+' operator.

- str1.charAt(9) - Retrieve the 9th Indexed character from string.

- str1.length() - Finds length of the string

- str1.toUpperCase() /toLowerCase() - Converts str1 to Upper/Lower case

- str1.indexOf('o') - Retrieve the Index of first 'o' character

- str1.indexOf('o',3) - Retrieve the index of 2nd most 'o' character

- Integer.parseInt(str3) – Converts string to integer.

- (num).toString – Converts Integer variable to String

- substring(5) – Removes the first 5 characters from the string.

- str3.substring(2, 7) - Prints the string starting from 3rd character to 7th character

9 CONDITIONAL STATEMENTS

- if, if else and nested if else statements are useful to take the decision based on conditional match.
This is useful when you want to execute some part of code based on particular condition

- ***Simple If Statement***

Part of code will be executed only if specified condition returns true. If condition will return false then that code will be not executed.

```
- if (a<b) {
        System.out.println("Value Of a ("+a+") Is less than Value Of b("+b+")." );
}
```

Note: Above Output line can be written on the same line also.

- ***If else Statement***

If condition returns true then part of if block will be executed. If condition returns false then part of else block will be executed.

Example :

```
if (a>=b)
{
        System.out.println("Value Of a("+a+") Is Greater Than Or Equals To Value Of
b("+b+")." );
}else
{
        System.out.println("Value Of a("+a+") Is Less Than Value Of b("+b+")." );
}
```

- Nested If Else Statements
- You can use nested if else statement when you wants to check multiple conditions and take decision based on it.

```
if (c<a)
{
        System.out.println("Value Of c("+c+") Is Less Than Value Of a("+a+")" );
}else if (c>=a && c<=b)
{
        System.out.println("Value Of c("+c+") Is In Between Value Of a("+i+") And
Value Of b("+b+")" );
}else
{
        System.out.println("Value Of c("+c+") Is Greater Than Value Of b("+b+")" );
}
```

10 LOOPS

- Loops in Java are important in automation as they help in running the same condition multiple times

For Loop

There are 3 components inside the For Loop - 1) Variable Initialization, 2. Condition to Terminate and 3. Increments variable (sometimes decrements too).

```
for(int i=0; i<=3; i++){
        System.out.println("Value Of Variable i is " +i);
}
```

While loop
In a while loop the block of code which is written inside while loop will be executed till the condition of while loop remains true

Example of While loop:

```
int a = 0;
while(a<=3)
{
        System.out.println("Value Of Variable a Is "+a);
        a++;
}
```

Do While Loop
Same as while loop, do while loop will be executed till the condition returns true.
Example :

```
int j=0;
do{
        System.out.println("Value Of Variable j Is "+j);
        j=j-1;
}while(j>0);
```

There is one difference between while and do while loop.

While loop will check condition at the beginning of code block so It will be executed only if condition (while(a<=3)) returns true.

do while loop will check condition at the end of code block so It will be executed minimum one time. After 1st time execution, it will check the condition and if it returns true then code of block will be executed once more or multiple time.

11 ARRAYS

- Imagine you need to store around 20 to 25 values. Since we learnt Datatypes now we understand that we can define 20 to 25 variables and store those values. But managing 25 variables will be a big overhead and not a good practice. This is where Arrays come into picture

- Arrays are helpful in storing multiple values of same data type (int, String or char) at the same time and each stored data location has unique index

- ***One Dimensional Array***

Array Index	0	1	2	3
Array Values	10	20	30	40

So in this case value 10 has index 0, value 20 has index 1, value 30 has index 2 etc. Thus each value can be identified by its index.

- Int i[] = new int[5] - This is how we declare an array with a length of 5

- Initialize each index of an array with a value. We will refer the above table for values

```
i[0] = 5;
i[1] = 10;
i[2] = 15;
i[3] = 20;
i[4] = 25;
```

- Print a particular index - System.out.println(i[1]);
This will print the value of index 1 i.e. 10

- Now if we want to print all the values of an array then we can put a for loop as below

```
for(int f=0; f<i.length; f++){
        System.out.println(a[f]);
}
```

- ***Two Dimensional Array***
- 2 Dimensional Array can be viewed as rows and columns like an Excel
- Define a 3 rows and 2 columns data as below

```
String str [][] = new String[3][2];
str[0][0]="John";
str[1][0]="Radha";
str[2][0]="Mark";
str[0][1]="john1";
str[1][1]="radha1";
str[2][1]="mark1";
```

- To Print ‘John’ - System.out.println(str[0][0]);
- To Print all values

```
for(int a=0; a<str.length; a++)
{
	for(int b=0; b<str[a].length; b++)
	{
		System.out.println(str[a][b]);
	}
}
```

12 METHODS IN JAVA

- Methods are useful when we need to perform actions which are repeatable in nature or in other words actions need to be performed multiple times. This is where Methods can be used.

- Methods are group of statements which is created to perform some actions or operation when your java code call it from main method.

```
public static void main(String[] args) {
        Test1(); //Test1() method called inside main method.
}

public static void Test1() // method called from main method
{
        for(int a=0;a<=3;a++)
        {
                System.out.println("Value of a is " +a);
        }
}
```

13 RETURN TYPE OF METHODS

- Any method can have return types like int, String, Boolean, etc.

- void means method is not returning any value.

```
static int c;
public static void main(String[] args) {
        Add(2,3);
        System.out.println("Value of c Is "+c);
}

public static int Add(int a, int b){
        c=a+b;
        return c;
}
```

14 STATIC AND NON-STATIC METHODS

- Method can be static or non-static.

- We can call static methods directly while we cannot call non static methods directly. You need to create and instantiate an object of class for calling non static methods.

- Rules for static methods and variables

> Can access static methods directly inside static method
> Cannot access non-static methods directly inside static method
> Can access static variables directly inside static method
> Cannot access non static variables directly inside static method

15 ACCESS MODIFIERS

- Defines the Visibility of methods

- public: Methods or variables from all class of same package or other package can be accessed.

- private: Methods or variables can be accessed only from same class from same package

- protected: Methods or variables can be accessed from classes of same package or sub classes of that class.

- No Access Modifier: Methods or variables can be accessed inside all class of same package only.

16 ABOUT ATA

Agile Testing Alliance (ATA) is a non-profit testing community and certification organization, created to grow agile testing awareness, practices and acceptance. ATA is a global alliance of visionary industry leaders, prominent authors, leading educational institutions and testing evangelists who are passionate in proliferation of agile in testing. There is a huge need of agile testing talent and ATA is a step towards filling that void. Our mission is to create a learning roadmap specifically in agile testing space. We understand that learning never stops and that testing community needs recognition in this quest for knowledge.

Hence we have mapped the journey with milestones that can be evaluated, certified and thus recognized.

Here is a snapshot of the learning roadmap

Our Social Media Presence is as below

Website: http://www.agiletestingalliance.org
Twitter handle @**AgileTA**
Facebook Page: https://www.facebook.com/AgileTestingAlliance
Youtube link: https://www.youtube.com/user/AgileTestingAlliance
LinkedIn profile: http://www.linkedin.com/groups/Agile-Testing-Alliance-5131844
SlideShare: Learning and sharing Presentations and information
http://www.slideshare.net/AgileTestingAlliance/
http://www.slideshare.net/ATASlides/