TEKNOTURF

INTRODUCTION TO JAVA

OVERVIEW

TEKNOTURF

JAVA is an open source technology for developing platform independent application.

In this module you will learn…

TEKNOTURF

- What is Java?
- Features of Java
- About JDK and JRE
- Writing a Program in java

Introduction to JAVA

TEKNOTURF

Developed by James Gosling at Sun Micro Systems

Introduced in 1995

Is one of the fastest growing programming languages of all times

Features of JAVA

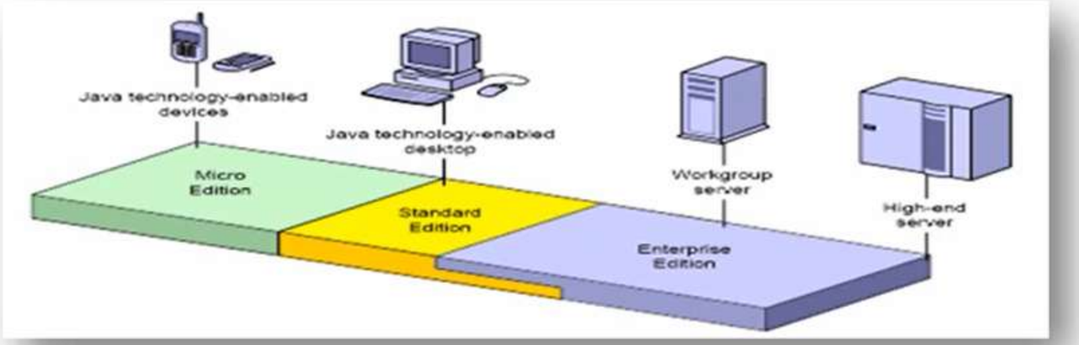
TEKNOTURF

- Object Oriented Programming Language
- Platform independence
- Architecture Neutral
- Secure
- Robust
  - Memory management(Automatic Garbage Collection)
  - Array limit checking
  - No direct access to memory addresses
  - Easy to create threads for multi processing
- Java can run:
  - in a browser as an Applet.
  - on the desktop as an Application.
  - in a server to provide Services e.g. Database access.
  - embedded in a device.

JAVA Editions

TEKNOTURF

- Java Standard Edition (J2SE or JSE)
  - Basic tools and libraries to build applications and applets.(Standalone applications)
- Java Enterprise Edition (J2EE or JEE)
  - Tools and libraries to create servlets , javaServerPages ,etc. (web applications)
- Java Micro Edition (J2ME or JME)
  - Libraries to create application to run on hand-held devices such as PDAs and cellular phones. (device programs and mobile applications)



Java Platform

TEKNOTURF

The **Java platform** is the name given to the computing platform from Oracle that helps users to *run* and *develop* Java applications.

The platform consists of two essential softwares:

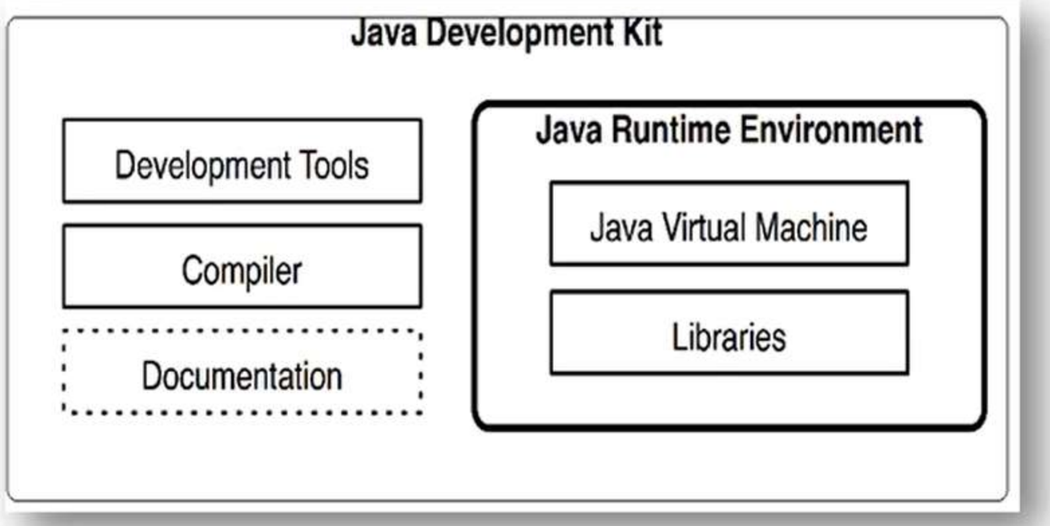
- the **Java Development Kit (JDK)**, which is needed to *develop* Java applications and applets
- the **Java Runtime Environment (JRE)**, which is needed to *run* Java applications and applets

JAVA Toolkit - JDK

TEKNOTURF

A Java Development Kit (JDK) is a program development environment for writing Java applets and applications.

JDK consists of a runtime environment called JRE, which sits on top of the operating system and the tools like interpreter (java) and compiler (javac).



JAVA Compiler

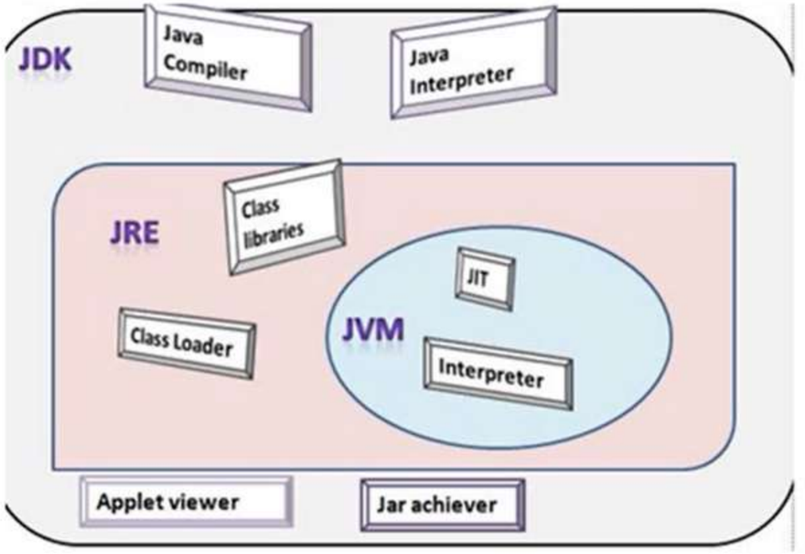
TEKNOTURF

- Java programs are translated into an intermediate code called byte code with the help of a java compiler
- Byte code is platform independent and can run on any operating system

JAVA Run-Time Environment - JRE

TEKNOTURF

- The Java Runtime Environment (JRE), also known as Java Runtime, is a part of the Java Development Kit (JDK)
- The Java Runtime Environment consists of the Java Virtual Machine (JVM), core classes, and supporting files.
- JVM converts Java byte code into machine language and executes it.



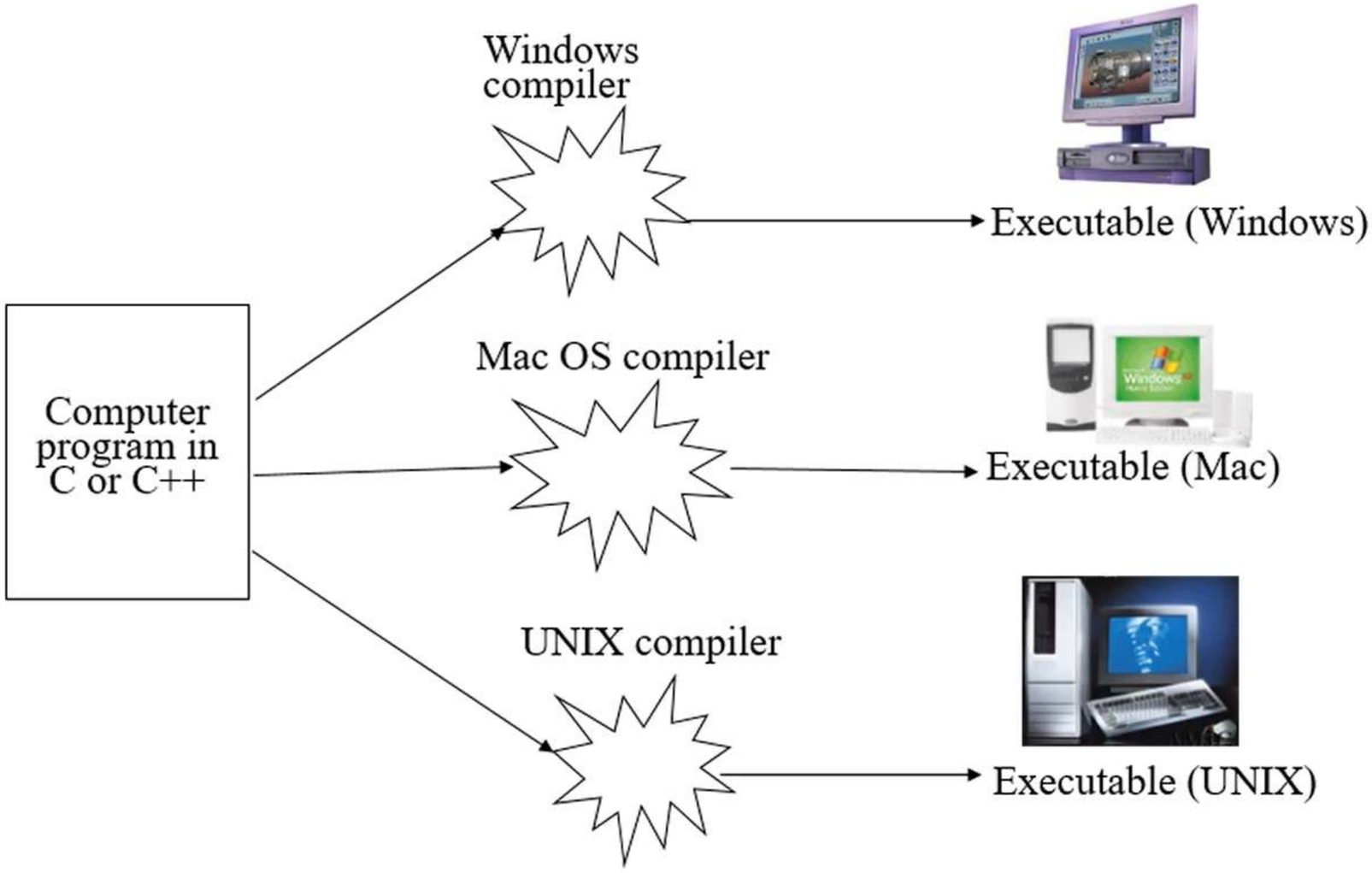
Platform Independent

TEKNOTURF

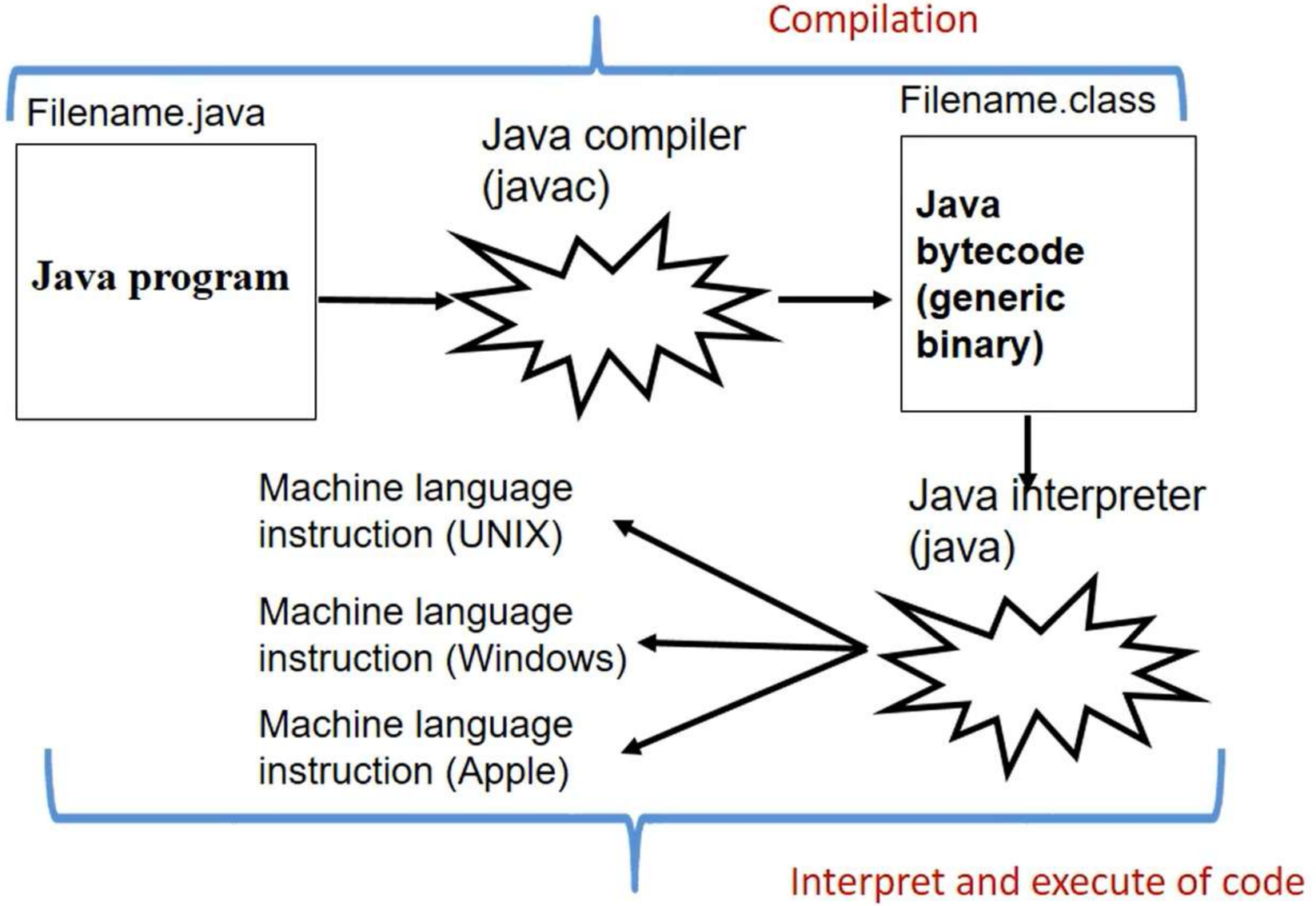
- The main feature of Java is platform independent

- The generated byte code which is the intermediate code is independent of the operating system and hence we say it is platform independent.

Platform Independent



Platform Independent



Write Once; Run Anywhere

Mac user running Safari

Click on link to Applet

Byte code is downloaded

Web page stored on Unix server

Virtual machine translates byte code to native Mac code and the Applet is run

Windows user running Internet Explorer

Write Once; Run Anywhere

Mac user running Safari

Windows user running Internet Explorer

Click on link to Applet

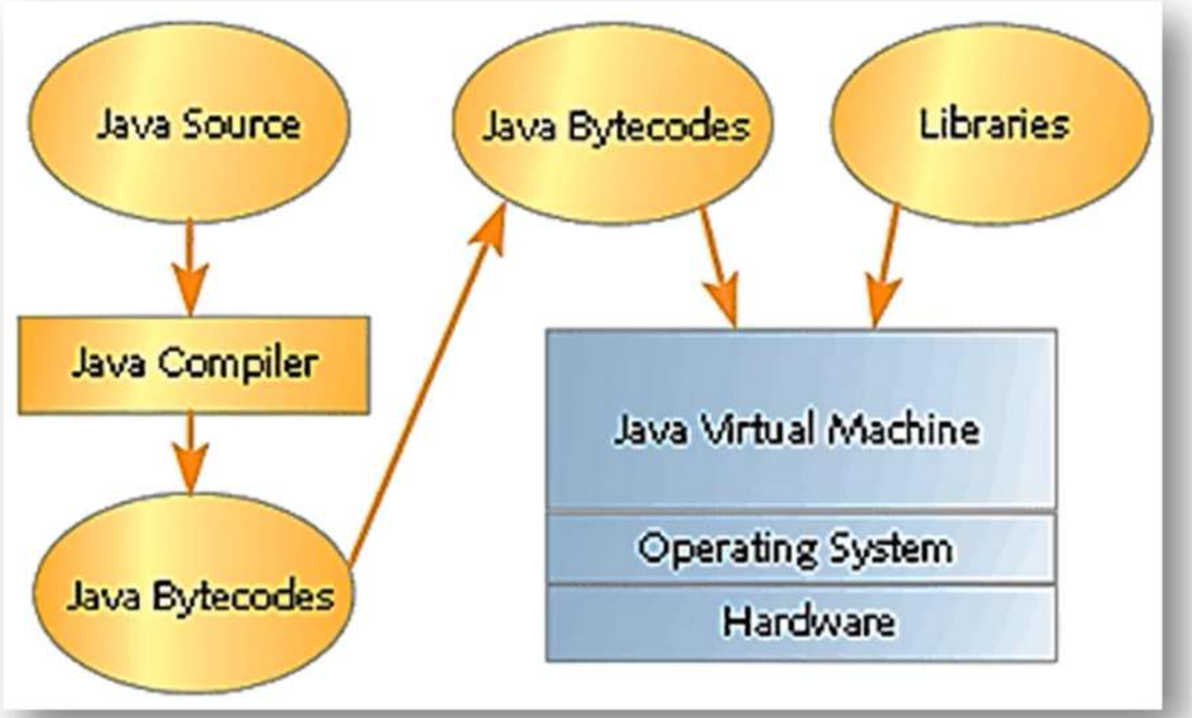
Byte code is downloaded

Web page stored on Unix server

Virtual machine translates byte code to native Windows code and the Applet is run

JAVA Program Execution

TEKNOTURF



Writing a Program in JAVA

TEKNOTURF

Any code written in java is written within a block termed as class

The main method in a java program acts as an entry point signature

All java programs start their execution from the main method

Main method

- has to be declared as public static
- takes String[] as an argument
- return type is void

Writing a Program in JAVA

TEKNOTURF

- Example : *Type the code in a notepad*

```
public class  TestProgram {
        public static void main(String a[]) {
                System.out.println("Welcome to Java");
        }
```

- Save this file with an extension .java
- Name of the file should be same as class name

TestProgram.java

Writing a Program **in JAVA**

TEKNOTURF

To compile and execute the code

- Click "**run**"
- Type "**cmd**"
- Get to the location where the file is located, say Desktop

  …/Desktop>
- Compile the code as

  …/Desktop> **javac** TestProgram.java
- To execute the code

  …/Desktop> **java** TestProgram
- Output will be

  Welcome to Java

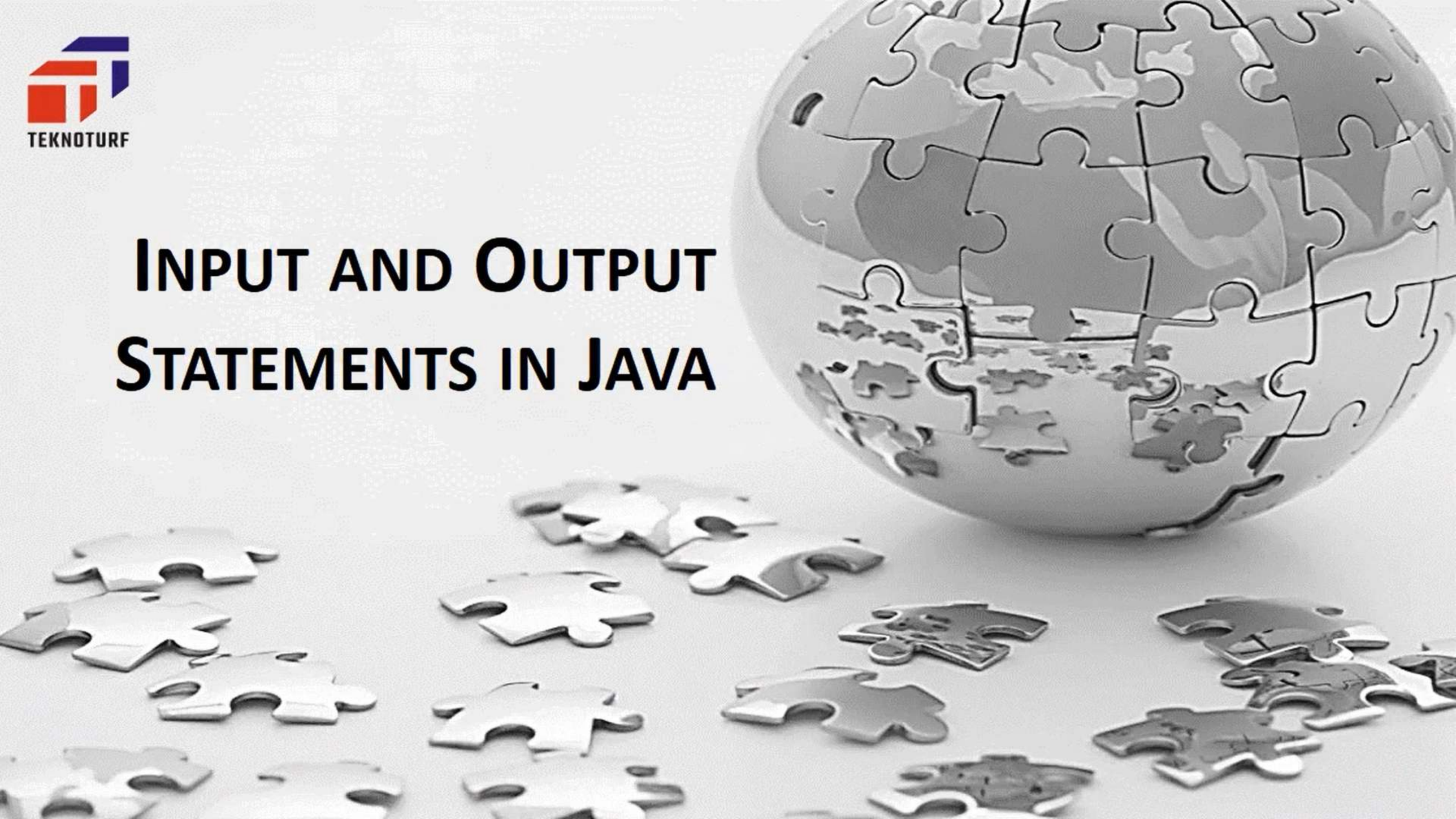
Note : when the code is compiled, it generates the intermediate code TestProgram.class, called the bytecode

Summary

- What is Java?
- Features of Java
- About JDK and JRE
- Writing a Program in java

TEKNOTURF

Input and Output Statements in Java



IN THIS MODULE YOU WILL LEARN

TEKNOTURF

- Print output using System.out
- Display output in formatted manner
- Get input from user using Scanner

OUTPUT STATEMENT

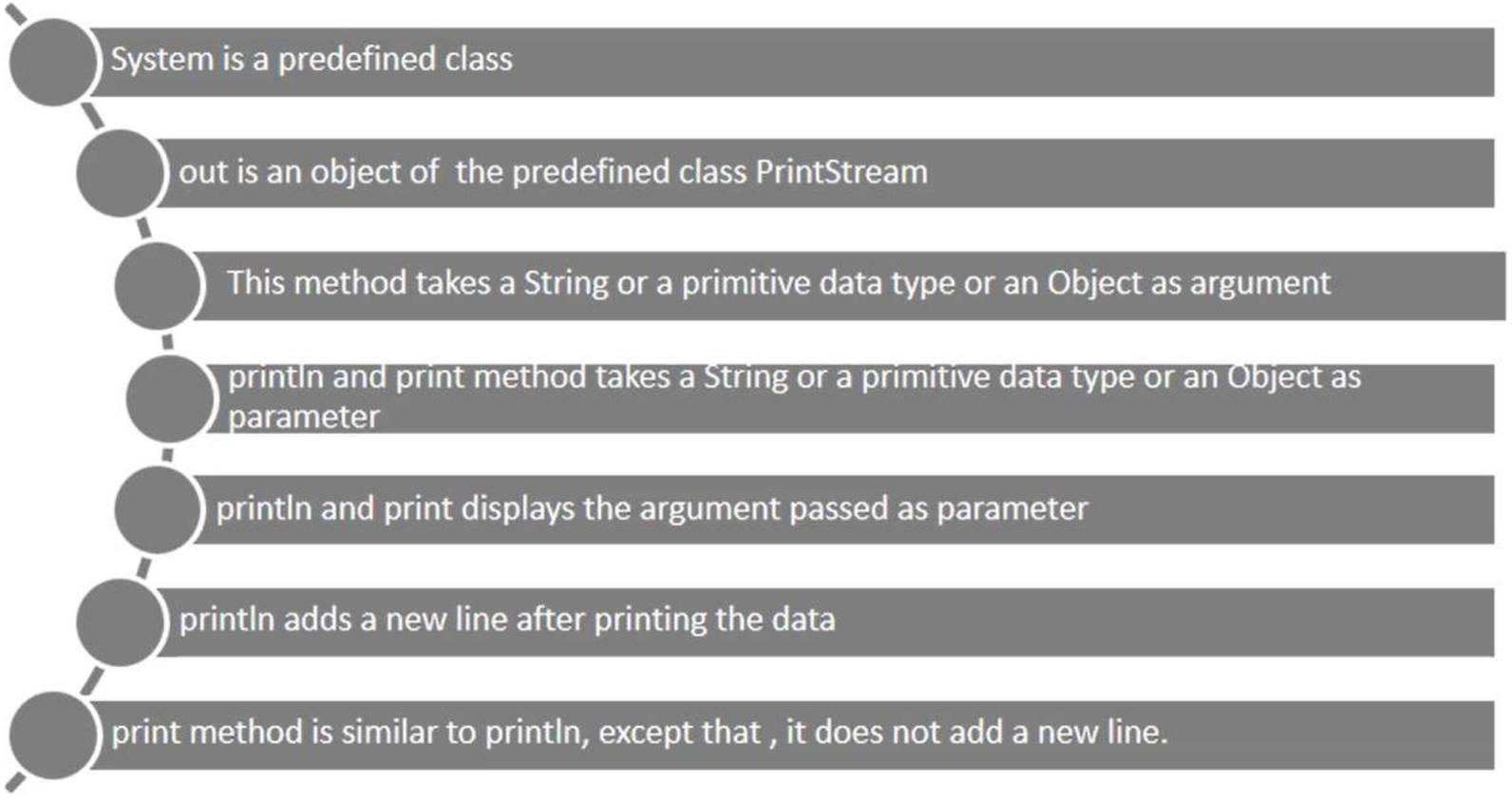
TEKNOTURF

Various ways to display output on the monitor Using system.out

System.out.println()

System.out.print()

System.out.printf(y)



USAGE OF PRINTLN VS PRINT

TEKNOTURF

When using println, new line is added

```
public class Test
{
    public static void main(String args[])
    {
        String name = "Peter";
        int id=1001;
        double salary = 85250.75;

        System.out.println("Welcome "+name);
        System.out.println("ID is "+id);
        System.out.println("Your salary is "+salary);

    }
}
```

Output

Welcome Peter
ID is 1001
Your salary is 85250.75

When using print, data is printed in the same line i.e...new line is not added

```
public class Test
{
    public static void main(String args[])
    {
        String name = "Peter";
        int id=1001;
        double salary = 85250.75;

        System.out.print("Welcome "+name);
        System.out.print("ID is "+id);
        System.out.print("Your salary is "+salary);

    }
}
```

Output

Welcome PeterID is 1001Your salary is 85250.75

USAGE OF ESCAPE SEQUENCES

TEKNOTURF

Escape Sequences

A character preceded by a backslash (\) is an escape sequence and has special meaning to the compiler.

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a form feed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

USAGE OF ESCAPE SEQUENCES

TEKNOTURF

Sample Code for Usage of common escape sequences like \n, \t, \' and \"

```
public class Test
{
    public static void main(String args[])
    {
        String name = "Peter";
        int id=1001;
        double salary = 85250.75;
        double nettSalary=salary-salary*0.15;

        System.out.println("Welcome \""+name+"\"");
        System.out.print("ID is \'"+id+"\'.\nYour salary is "
        +salary+".\tYour nett salary is "+nettSalary);

    }
}
```

Output

Welcome "Peter"
ID is '1001'.
Your salary is 85250.75. Your nett salary is 72463.1375

PRINTF STATEMENT

TEKNOTURF

PRINTF METHOD USED TO DISPLAY OUTPUT IN A FORMATTED MANNER

Example : To print the net salary of an employee with multiple digits in decimal to 2 decimal places, requires formatting which needs to be mentioned in the format

Syntax : System.out.printf(format, arguments)

Specify the formatting rules using the format specifier.

Rules start with the '%' character.

The format specifiers for general, character, and numeric types have the following syntax:

%[argument_index$][flags][width][.precision]conversion-character

Here except the conversion-character all the rest are optional

PRINTF STATEMENT

TEKNOTURF

The [argument_index] is a decimal integer that specifies the position of the argument in the argument list. The first argument is referenced by "1$", the second by "2$", etc.

The [flags] define standard ways to modify the output and are most common for formatting integers and floating point numbers

The [width] specifies the field width for outputting the argument. It represents the minimum number of characters written to the output

The [.precision] specifies the number of digits of precision when outputting floating-point values. Additionally, we can use it to define the length of a substring to extract from a String.

The conversion-character determines how the argument is formatted. Conversion characters are only valid for certain data types. Some common characters are :

- s - formats strings
- d - formats decimal integers
- f - formats the floating-point numbers
- c – formats character
- t - formats date/time values
- n – line separator

PRINTF STATEMENT

TEKNOTURF

Example :

```
public class Test
{
    public static void main(String args[])
    {
        String name="Raghu";
        int id = 1001;
        float salary = 72463.1375f;
        boolean status = true;

        //%2$ - Here 2 specifies the 2nd argument name, %3$,.2f – 3$ means 3rd argument salary
        //"," specifies the flag, ".2f" specifies correct to 2 decimal places
        System.out.printf("%2$s %3$,.2f %1$d %b",id,name,salary,status);

        System.out.printf("%n%d %S %,.2f %B",id,name,salary,status);

        System.out.printf("\n\nID is %d %nName is %s. Salary is %.2f",id,name,salary);
    }
}
```

Output

Raghu 72,463.14 1001 true
1001 RAGHU 72,463.14 TRUE

ID is 1001
Name is Raghu.
Salary is 72463.14

STRING FORMATTING

TEKNOTURF

- %s is the format specifier for a simple string

- To specify a length for the string, a width can be specified

  Example : ***System.out.printf("%15s","Welcome");***

  Output : ‘ Welcome’

- To left justify the string, we can use ‘-’ flag

  Example : ***System.out.printf("%-15s","Welcome");***

  Output : ‘Welcome ’

STRING FORMATTING

TEKNOTURF

Example :

```
public class Test
{
    public static void main(String args[])
    {
        String name="Raghu";
        int id = 1001;
        float salary = 72463.1375f;

        System.out.printf("%-5s %-15s %-15s\n","ID","Name","Salary");

        //Convert primitive to String using String.valueOf
        String str1 = String.format("%-5s %-15s %-15s",String.valueOf(id),name,String.format("%.2f",salary));
        System.out.println(str1);


        System.out.printf("%-5s %-15s %-15s\n","ID","Name","Salary");
        //Alternate way to convert primitive to String
        String str2 = String.format("%-5s %-15s %-15s",id+"",name,""+String.format("%.2f",salary));
        System.out.println(str2);
    }
}
```

Output

```
ID    Name          Salary
1001  Raghu          72463.14

ID    Name          Salary
1001   Raghu         72463.14
```

STRING FORMATTING

TEKNOTURF

Example :

```
public class Student {

    private int studentId;
    private String name;
    private float cgpa;
    private float tutionFees;

    public Student(int studentId, String name, float cgpa, float tutionFees) {
        this.studentId = studentId;
        this.name = name;
        this.cgpa = cgpa;
        this.tutionFees = tutionFees;
    }

    public String toString(){
        return String.format("%-15s %-20s %-5s %-15s",String.valueOf(studentId),
                name,String.valueOf(cgpa),String.valueOf(tutionFees));
    }

    public static void main(String arg[]){
        Student s1=new Student(101,"Raghu",9.7f,30000);
        System.out.printf("%-15s %-20s %-5s %-15s\n","Student ID","Name","CGPA","Fees");
        System.out.println(s1);
    }
}
```

Output

Student ID	Name	CGPA	Fees
101	Raghu	9.7	30000.0

READ INPUT FROM USER

TEKNOTURF

- In the real world, it is necessary for programs to get input from the user

  Input can be read from keyboard in many ways
- One such way is to use the inbuilt class Scanner
- Predefined / Inbuilt classes in Java are organized in the form of packages.

  Scanner class is present in java.util package
- To use Scanner class, include package java.util using import statement as :

  import java.util.Scanner ;
  // imports just the Scanner class

  or

  import java.util.*;
  // imports the entire package java.util

- Next create a Scanner object

  Scanner sc = new Scanner(System.in); // System.in indicates input will
  //be given to System from keyboard

```java
import java.util.Scanner;

public class Test
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter id");
        int id = sc.nextInt();
        System.out.println("Enter name");
        String name = sc.next();

        System.out.println(id+"  "+name);
    }
}
```

READ INPUT FROM USER

TEKNOTURF

Scanner class has various methods to get input of various data types from the user

Method	Data type
nextInt()	Integer
nextFloat()	Float
nextDouble()	Double
nextLong()	Long
nextShort()	Short
nextBoolean()	Boolean
next()	Single word (String without space)
nextLine()	String with spaces

```
import java.util.Scanner;

public class Test
{
    public static void main(String args[])
    {
        Scanner sc = new  Scanner(System.in);

        float salary = sc.nextFloat();
        int count = sc.nextInt();
        byte b1 = sc.nextByte();
        long phone = sc.nextLong();
        boolean status = sc.nextBoolean();
        double price = sc.nextDouble();
        String word = sc.next();
        String sentence = sc.nextLine();
    }
}
```

READ INPUT FROM USER

TEKNOTURF

READ STRING

String is a collection of characters.

next() method in Scanner reads a String, with only one word

Even if the input has multiple words, next() takes the first word alone as input

To get a String as input with spaces, use the nextLine() method.

```
import java.util.Scanner;

public class Test
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the name");
        String name = sc.nextLine();

        System.out.println("Enter the city");
        String city = sc.next();


        System.out.println("Name is "+name+". City is "+city);
    }
}
```

READ STRING USING NEXTLINE

TEKNOTURF

- ✓ Observe the code below
- ✓ On executing the above code, city will not be taken as input. Scanner skips the nextLine()

```
Scanner sc = new Scanner(System.in);

System.out.println("Enter the name");
String name = sc.next();

System.out.println("Enter the city");
String city = sc.nextLine();

System.out.println("Name is "+name+". City is "+city);
```

- ✓ Scanner skips nextLine() when it is used after next() or after any nextXXX() method.
- ✓ This is because next() reads only the String, not the newline (when Enter key is pressed) nextLine() consumes the nextline(\n) as input. Hence no input is received.
- ✓ Easy way to resolve :

  When using nextLine after next() or nextXXX(), consume the newline by including the statement sc.nextLine().

READ STRING USING NEXTLINE

TEKNOTURF

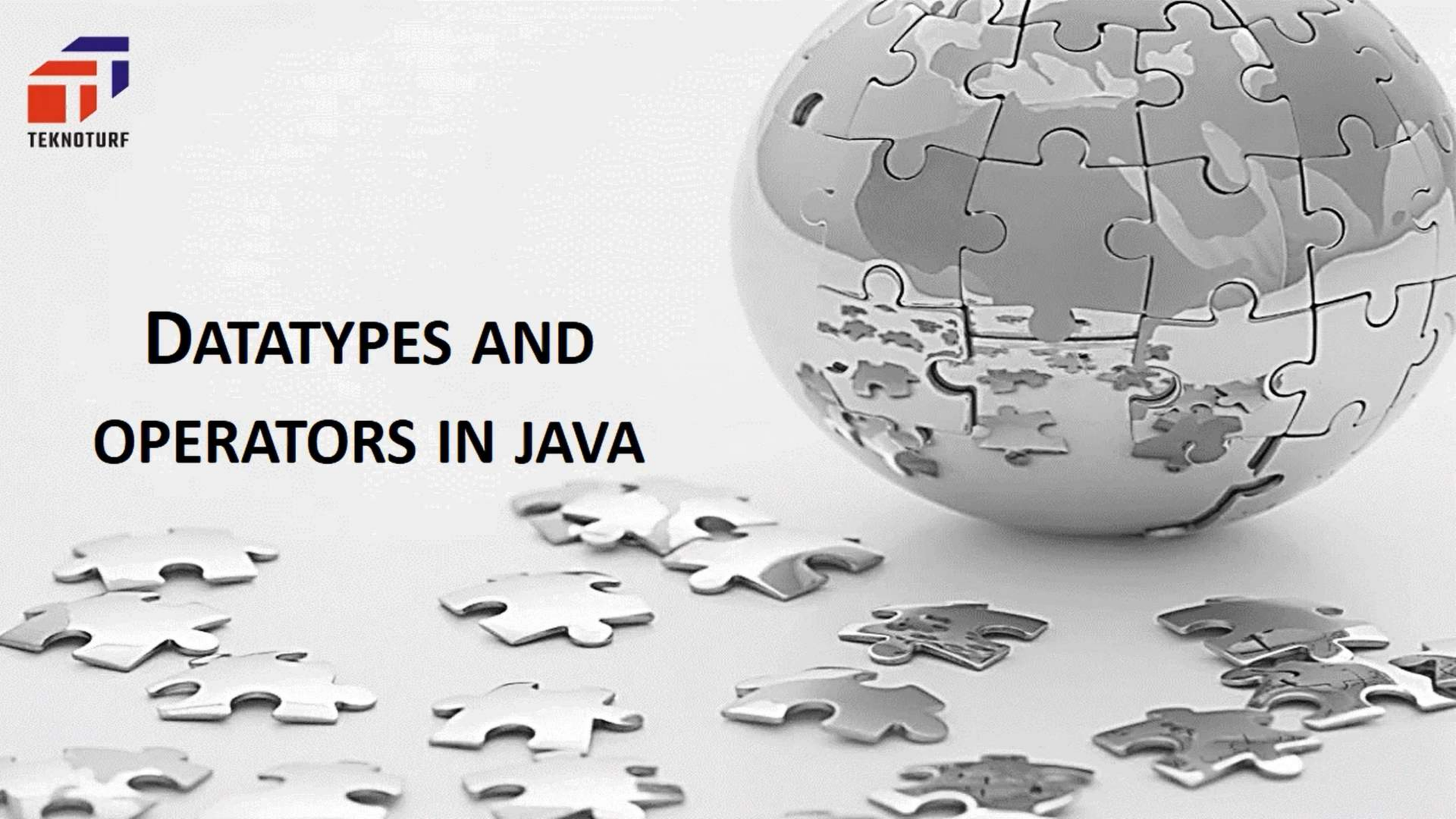
USING NEXTLINE() AFTER NEXT() OR NEXTXXX()

```
import java.util.Scanner;

public class Test
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the id");
        int id = sc.nextInt();
        sc.nextLine(); //captures the newline(Enter key)
        System.out.println("Enter the name");
        String name = sc.nextLine();

        System.out.println("Enter the salary");
        double salary = sc.nextDouble();
        sc.nextLine(); //captures the newline(Enter key)

        System.out.println("Enter the city");
        String city = sc.nextLine();

        System.out.println("Name is "+name+". City is "+city);
    }
}
```

- After getting the id using nextInt we have the name using nextLine.
- nextInt captures only the number and hence nextLine captures the new line.
- So it will skip getting input for name and get the input for salary.
- To solve this issue, we have included the statement sc.nextLine to capture the new line.
- Similarly we have obtained the input for salary and city.

TEKNOTURF

DATATYPES AND OPERATORS IN JAVA



In this module you will learn…

TEKNOTURF

- Datatypes
- Operators
- Precedence and Associativity of operators

Datatypes - Overview

TEKNOTURF

Assume David is a customer in ABC Bank.

The bank needs to create the customer with the following fields.

- Name
- Gender
- Account Number
- Balance
- Status

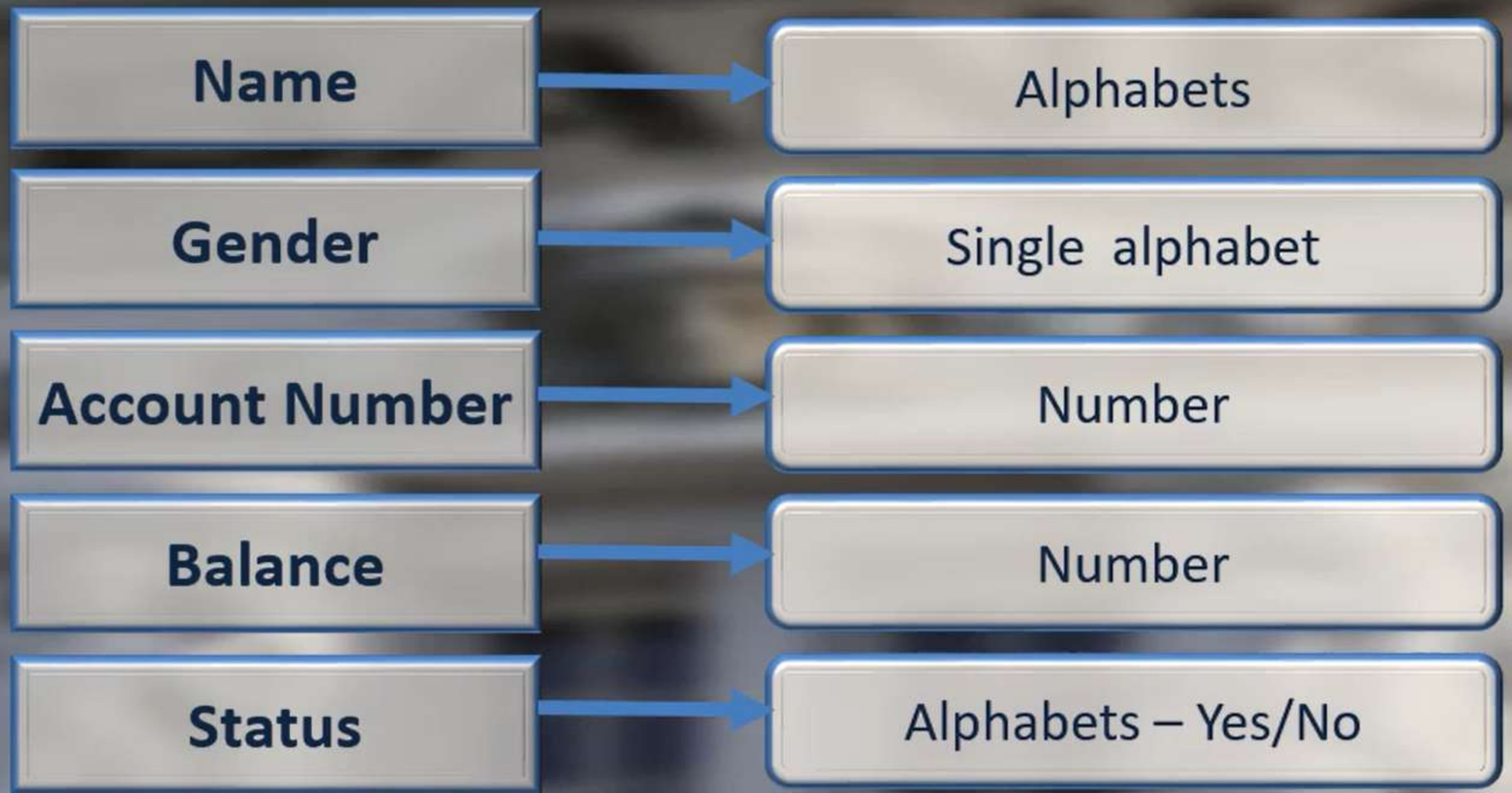
Each and every field will hold data of different types.

Datatypes - Overview

TEKNOTURF

Assume David is a customer in ABC Bank.

The bank needs to create the customer with the following fields.



Java Comments

TEKNOTURF

Java supports the following comment styles

Single line comment

```
// comment on one line
```

Multi line comment

```
/* comment on one
 * or more lines
*/
```

Java documentation comment

```
/** documentation comment
 *can also span one or more lines
*/
```

Semicolons, Blocks and White Space

TEKNOTURF

A statement is one or more lines of code terminated by a semicolon (;)

```
totals = a + b + c+ d + e + f;
```

A block is a collection of statements bound by opening and closing braces

```
{
    x = y + 1;
}
```

A class definition uses a special block:

```
public class MyDate {
        private int day;
        private int month;
        private int year;
}
```

Any amount of white space is permitted in a Java program.

Java Identifiers

TEKNOTURF

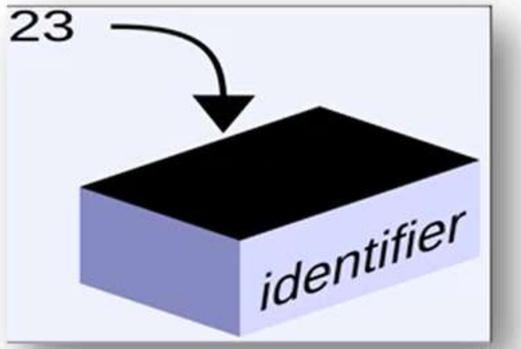
Identifiers are names given for class, variables , methods

Identifiers have the following characteristics:

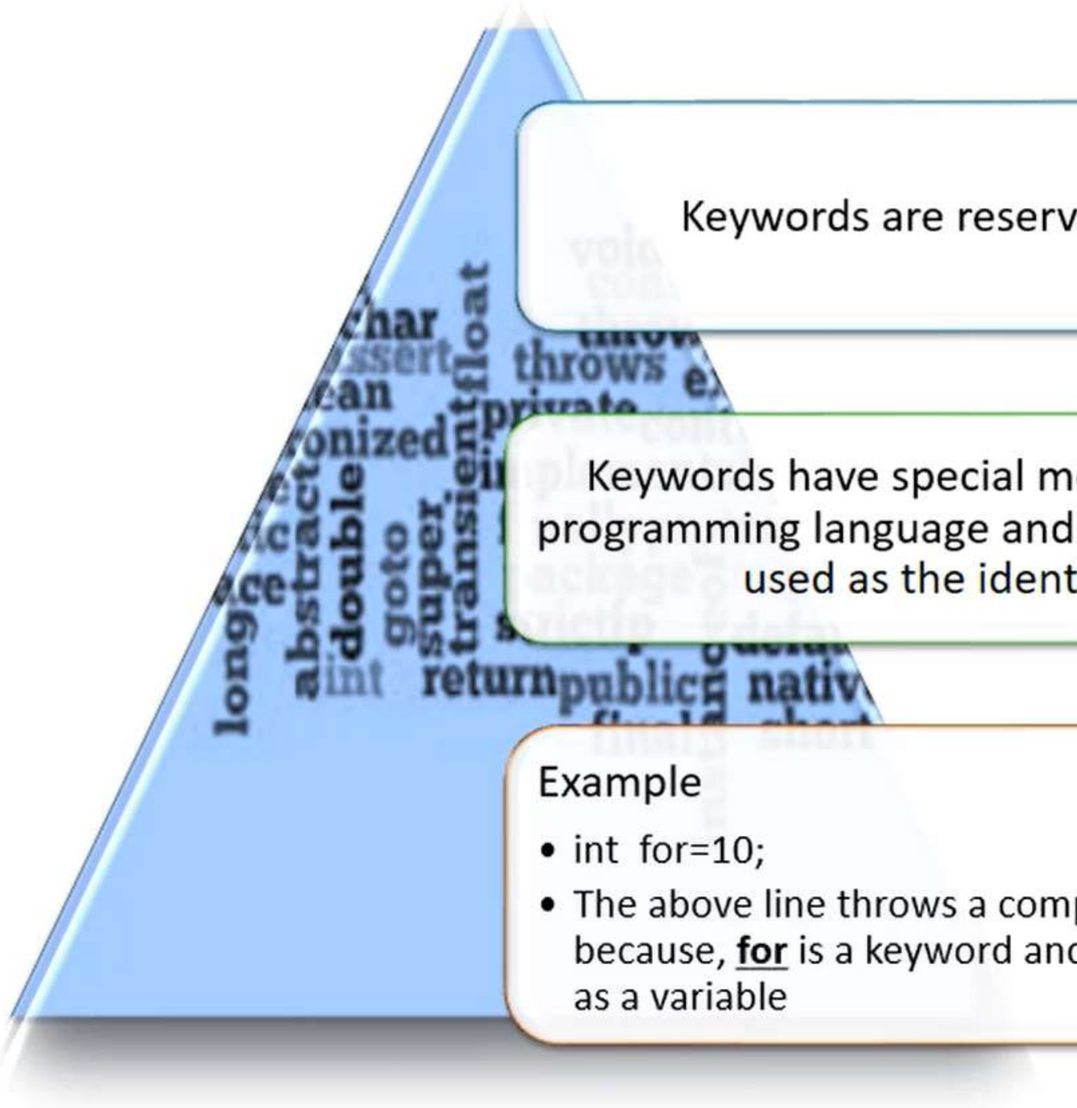
- Identifier names given to a variable, class, or method
- Identifier names can start with an alphabet, underscore (_), or dollar sign ($) and followed by alphabets / numbers
- Are case-sensitive and have no maximum length

Examples:

- identifier
- userName
- user_name
- _sys_var1
- $change



Java Language Keywords



Keywords are reserved words

Keywords have special meaning in the programming language and they cannot be used as the identifiers

Example

- int for=10;
- The above line throws a compile time error because, **for** is a keyword and cannot be used as a variable

Java Language Keywords

TEKNOTURF

Keywords in Java				
abstract	default	if	private	this
assert	do	implements	protected	throw
boolean	double	import	public	throws
break	else	instanceof	return	transient
byte	enum	int	short	try
case	extends	interface	static	void
catch	final	long	strictfp	volatile
char	finally	native	super	while
class	float	new	switch	
continue	for	package	synchronized	

Data Types

TEKNOTURF

Data type defines the set of values a variable can hold in its life time

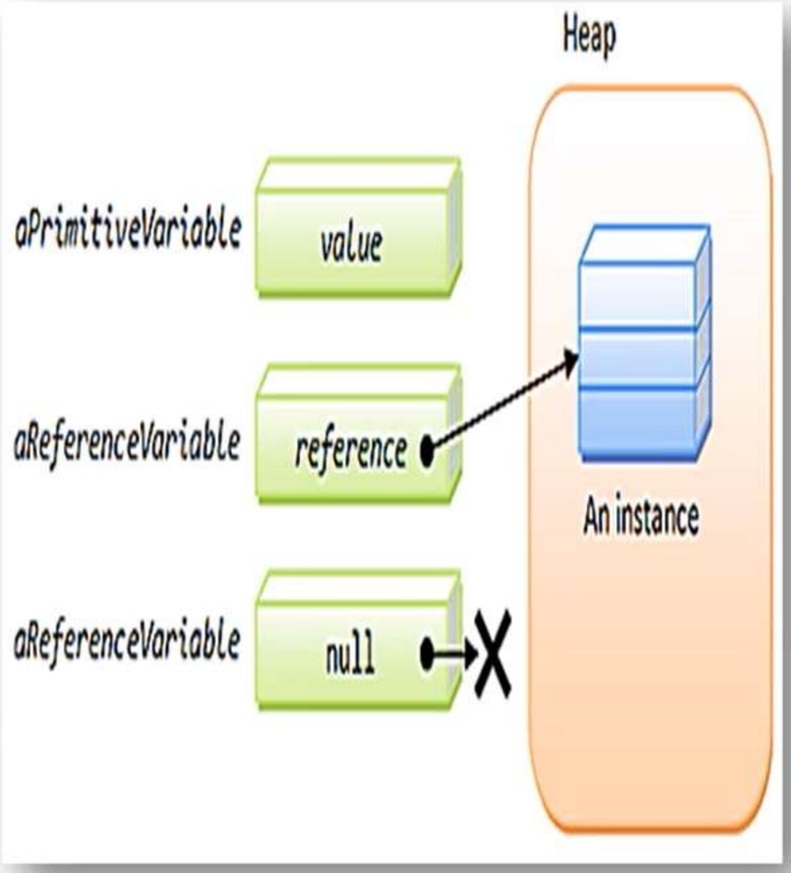
Java supports the following data type

Primitive data type

- Holds a scalar value
- int age=23;

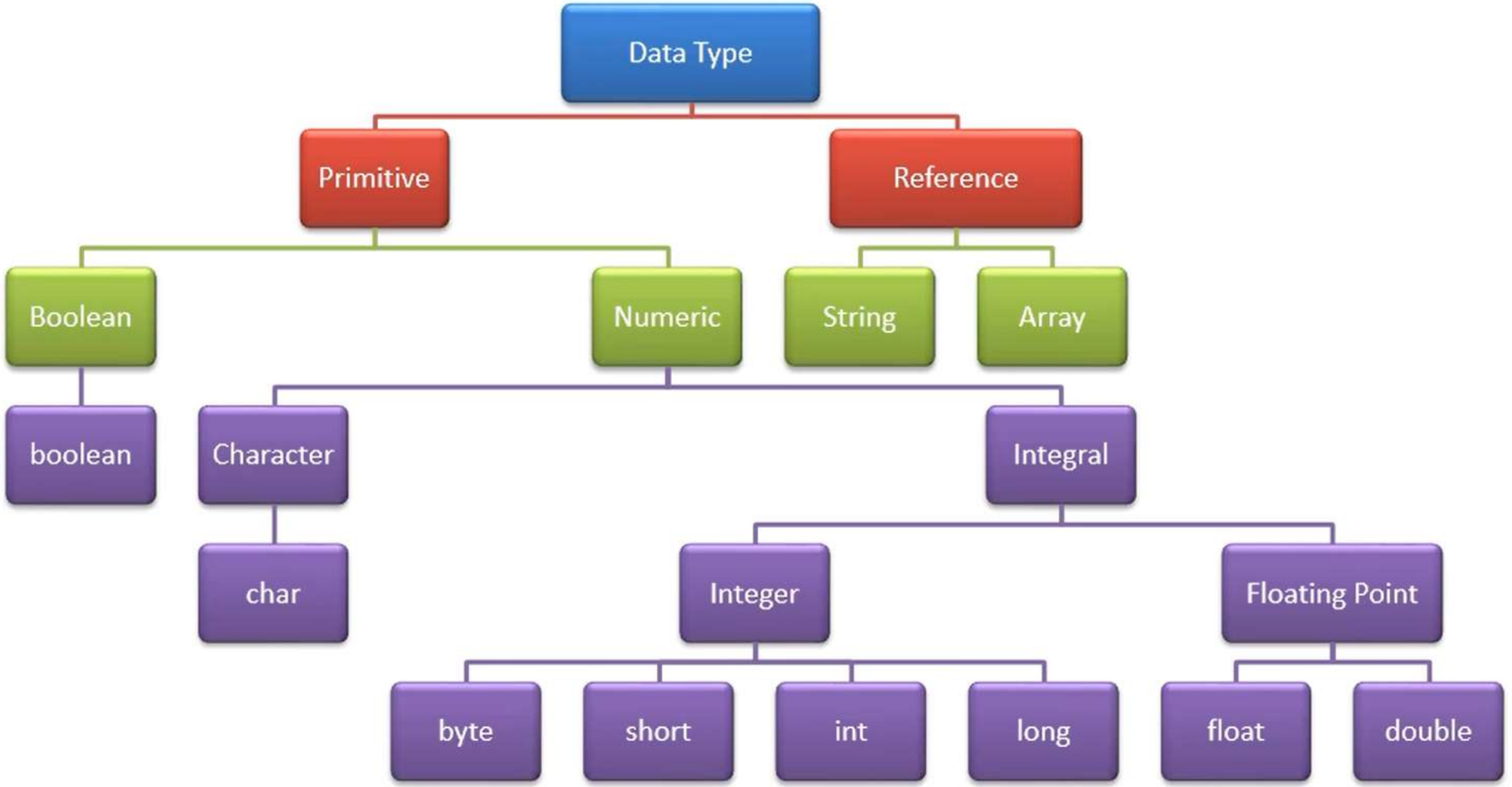
Reference data type

- Holds an object
- Employee emp = new Employee();
- emp is a reference holding an employee object
- All Objects are allocated memory in the heap



Data Types

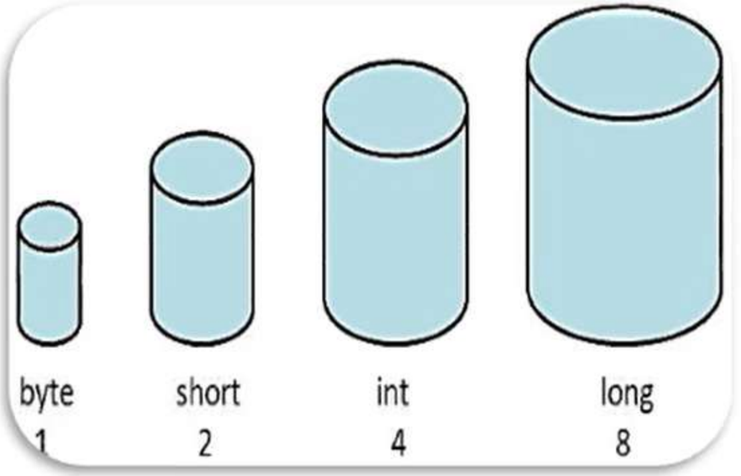
TEKNOTURF



Data Types

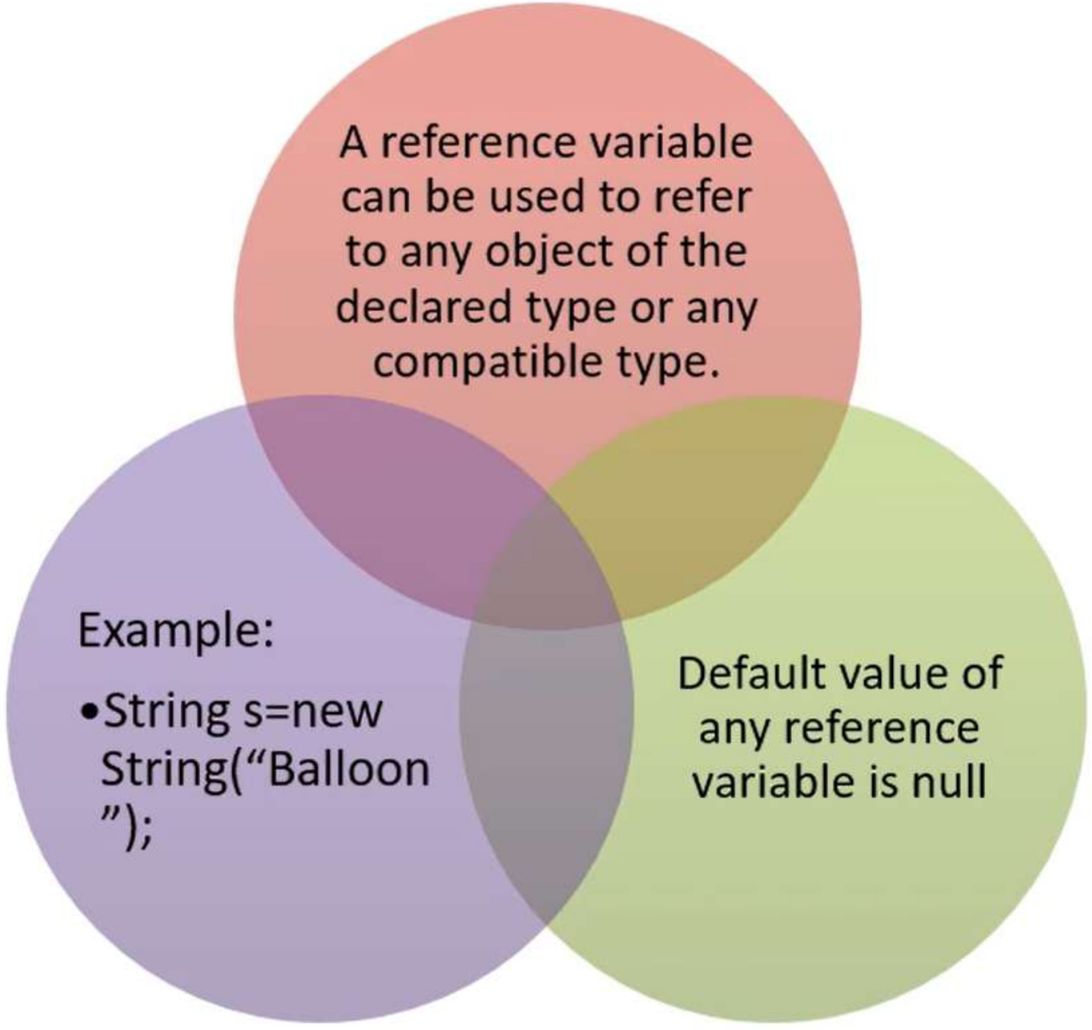
TEKNOTURF

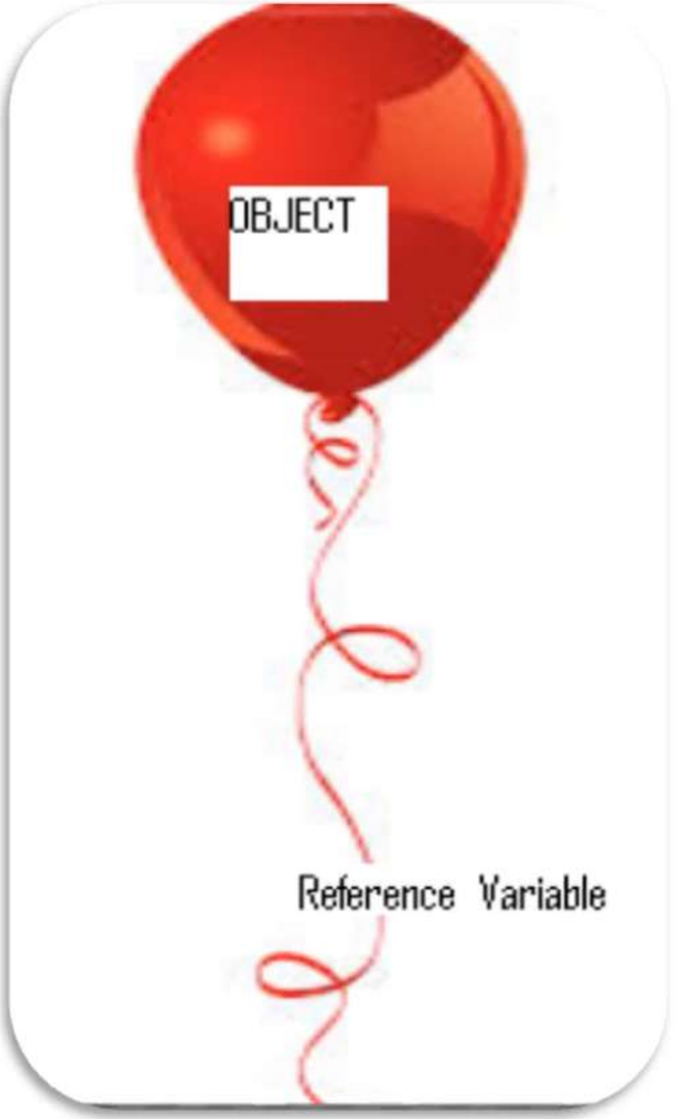
Data type	Size (in Bytes)	Default Value
byte	1	0
short	2	0
int	4	0
long	8	0
float	4	0.0
double	8	0.0
char	2(UNICODE)	\u0000
boolean	Depends on JVM	false



Reference Data Types

TEKNOTURF





Casting and Conversion

TEKNOTURF

Lower size data type can be assigned to higher size data type automatically .

- **Implicit conversion**

Example for conversion

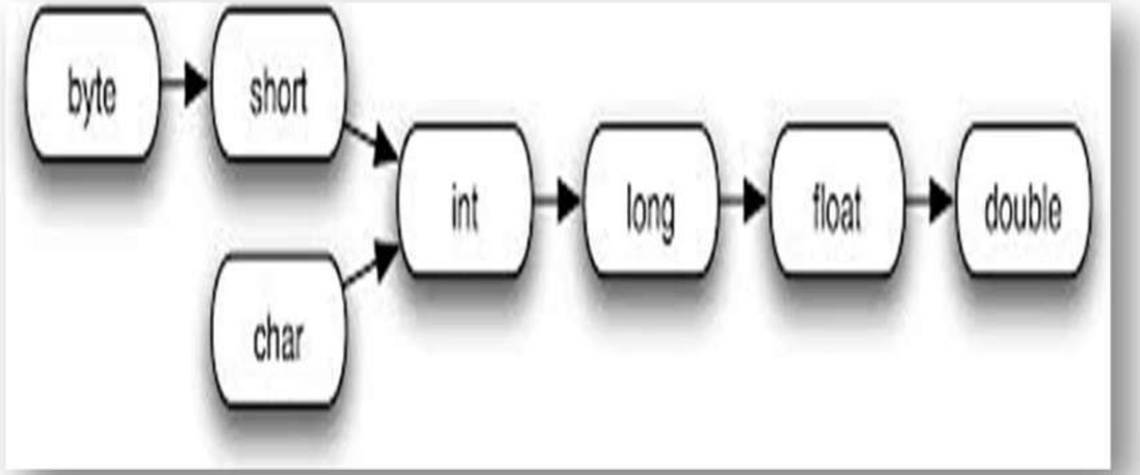
- **char c ='V';**
- **int i = c;**

Casting is the process of converting the value to different type than its source

- **Explicit conversion**

Example for Casting

- **long val =10;**
- **int x = (int) val;**



Operators - Overview

TEKNOTURF

Fields for a customer in a Bank

- Name
- Gender
- Account Number
- Balance
- Status

How to calculate the interest amount based on the balance available ?

To perform mathematical and logical manipulations on data we have to use **Operators.**

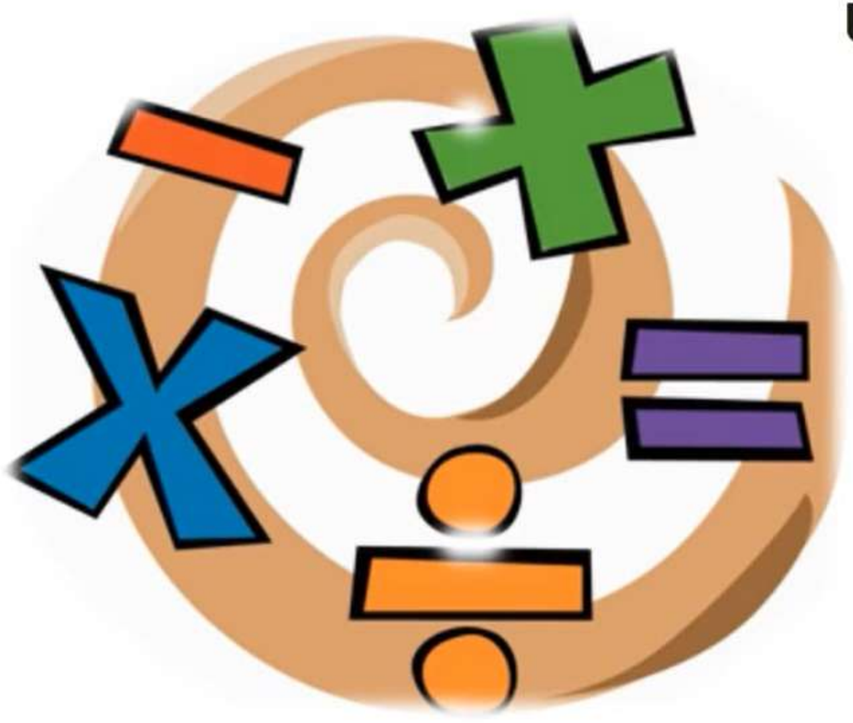
Types of Operators

TEKNOTURF

Arithmetic

Miscellaneous

Unary Arithmetic



Ternary

Relational and logical

Bitwise

Arithmetic and Unary Operators

TEKNOTURF

Operator	Description	Example(X=10,Y=20)
+	Addition – Adds values on either side of the operator	X + Y will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	X - Y will give -10
*	Multiplication - Multiplies values on either side of the operator	X * Y will give 200
/	Division - Divides left hand operand by right hand operand	Y / X will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	Y % X will give 0
++	Increment - Increases the value of operand by 1	Y++ gives 21
--	Decrement - Decreases the value of operand by 1	Y-- gives 19

Relational Operators

TEKNOTURF

Operator	Description	Example(A=5,B=10)
==	Checks if the values of two operands are equal or not; if equal, then condition results in true.	(A==B) is false.
!=	Checks if the values of two operands are equal or not; if values are not equal, then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand; if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand; if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then condition becomes true.	(A <= B) is true.

Bitwise Operators

TEKNOTURF

Operator	Description	Example (A=15,B=10)
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 10 which is 0000 1010
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 15 which is 0000 1111
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 5 which is 0000 0101
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -16 which is 1111 0000 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operand's value is moved left by the number of bits specified by the right operand	A << 2 will give 60 which is 0011 1100
>>	Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand.	A >> 2 will give 3 which is 0000 0011
>>>	Shift right zero fill operator. The left operand's value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 3 which is 0000 0011

Logical Operators

TEKNOTURF

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverse the logical state of its operand. If a condition is true then Logical NOT operator will make it false.	!(A && B) is true.

Assignment Operators

TEKNOTURF

Operator	Description	Example
=	Simple assignment operator; it assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator; it adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator; it subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator; it multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A

Assignment Operators

TEKNOTURF

Operator	Description	Example
/=	Divide AND assignment operator. It divides left operand with the right operand and assigns the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

Miscellaneous Operators

TEKNOTURF

Conditional Operator

- ?:
- variable x = (expression) ? value if true : value if false
- Example
  - int a=10,b=2;
  - b = a > 10?a:b;

instanceof operator

- This operator is used to check whether the object is of a particular type(class or interface)
- (Object reference variable) instanceof (class/interface type)
- String s=new String(“Balloon”);
- s instanceof String

Precedence and Associativity

TEKNOTURF

Priority of Operators determines the Precedence

Two Operators of equal priority in the expression is evaluated through associativity

Associativity will be either Left → Right or Right → Left

Precedence and Associativity

TEKNOTURF

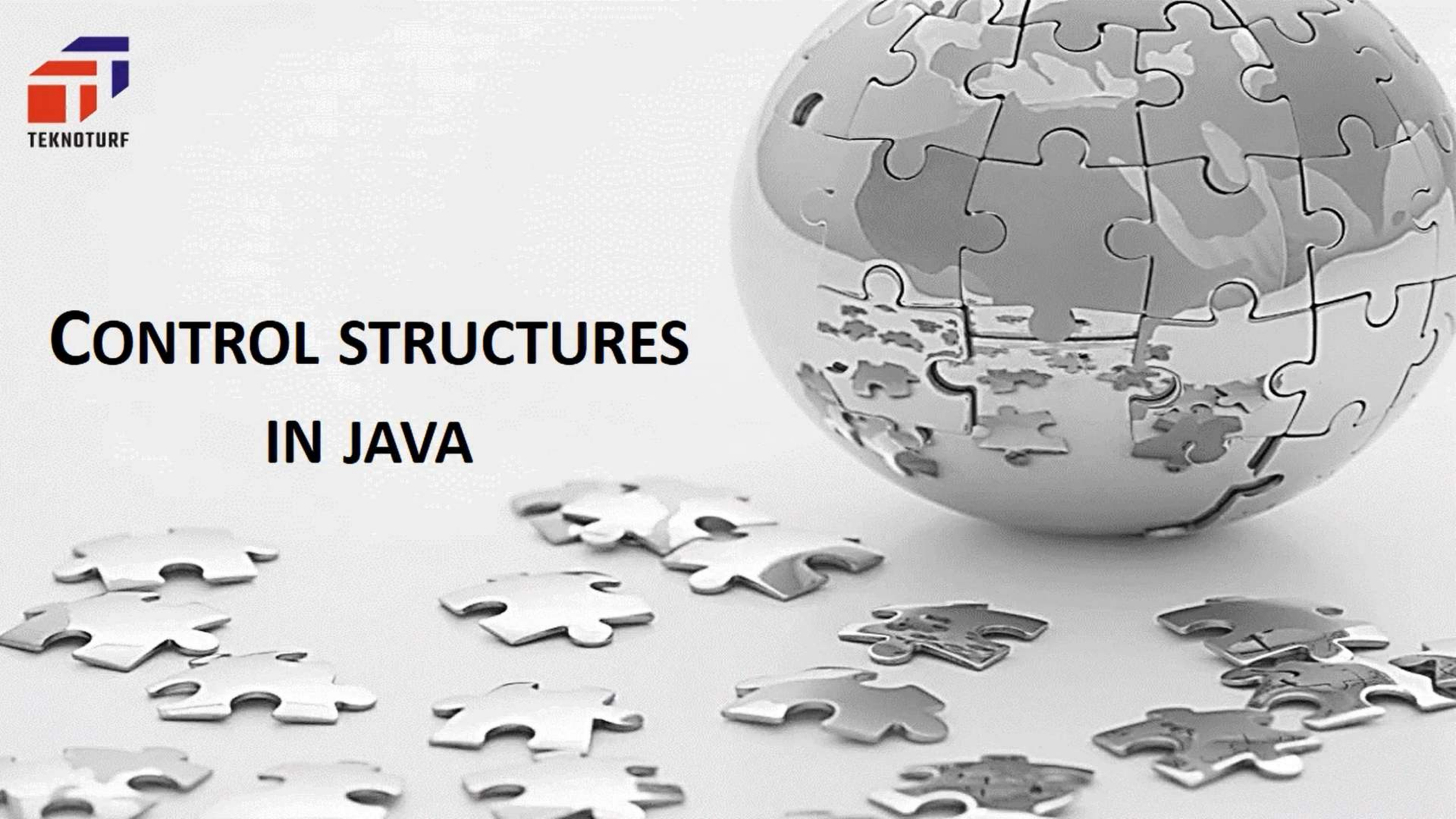
Category	Operator	Associativity
Postfix	() [] . (dot operator)	Left toright
Unary	++ - - ! ~	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	>> >>> <<	Left to right
Relational	> >= < <=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right

Summary

- Datatypes
- Operators
- Precedence and Associativity of operators

TEKNOTURF

CONTROL STRUCTURES IN JAVA



In this module you will learn…

TEKNOTURF

- Selection Statement
- Iterative Statement

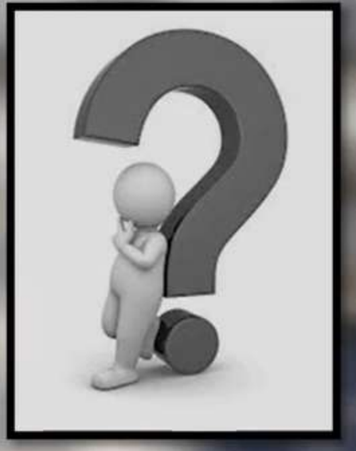
Control Structures - Overview

TEKNOTURF

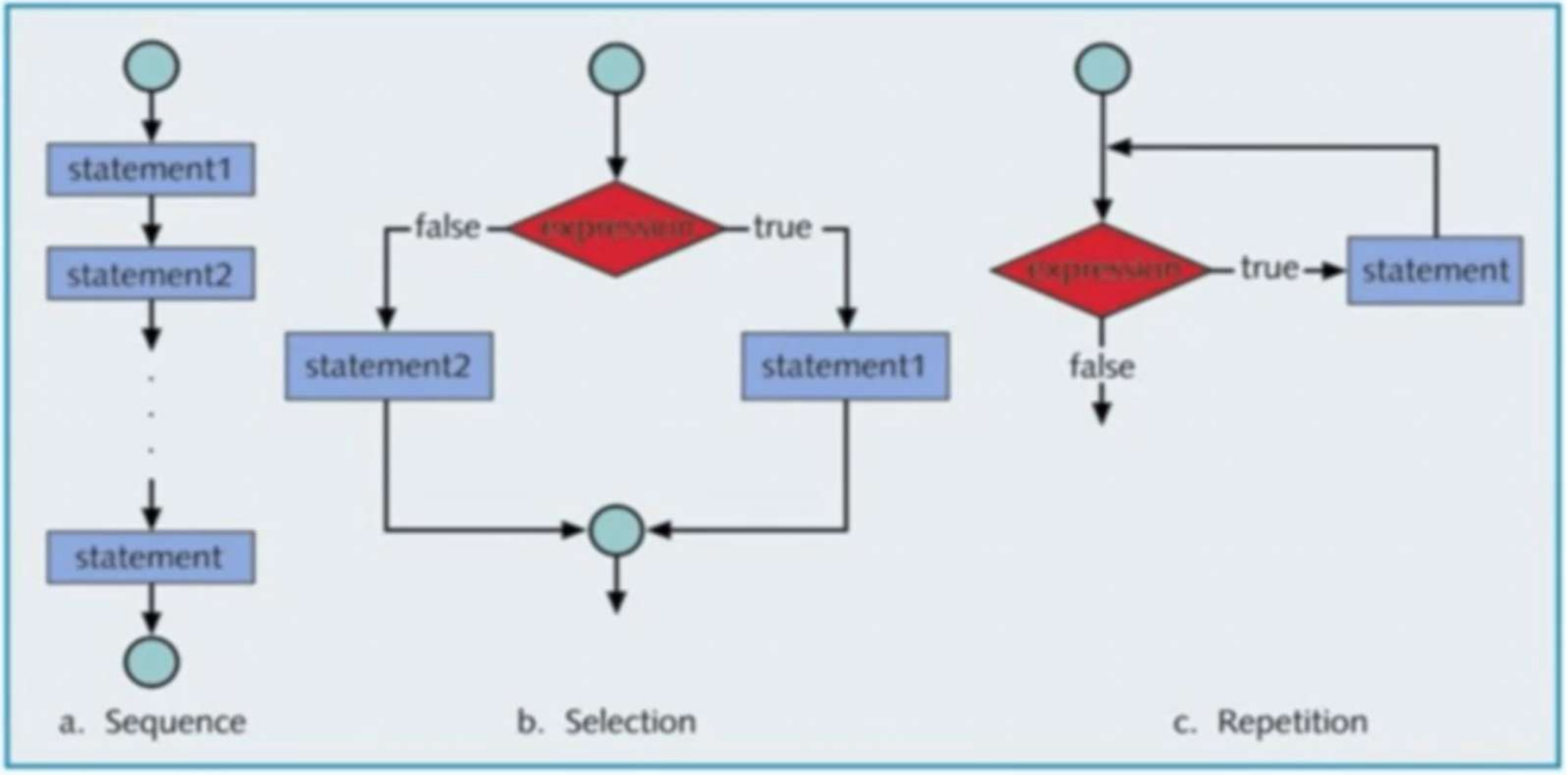
I am unable to withdraw amount from my account. What could be the reason?

To withdraw amount, your balance should be greater than the minimum balance.

How can this be done in an application ?

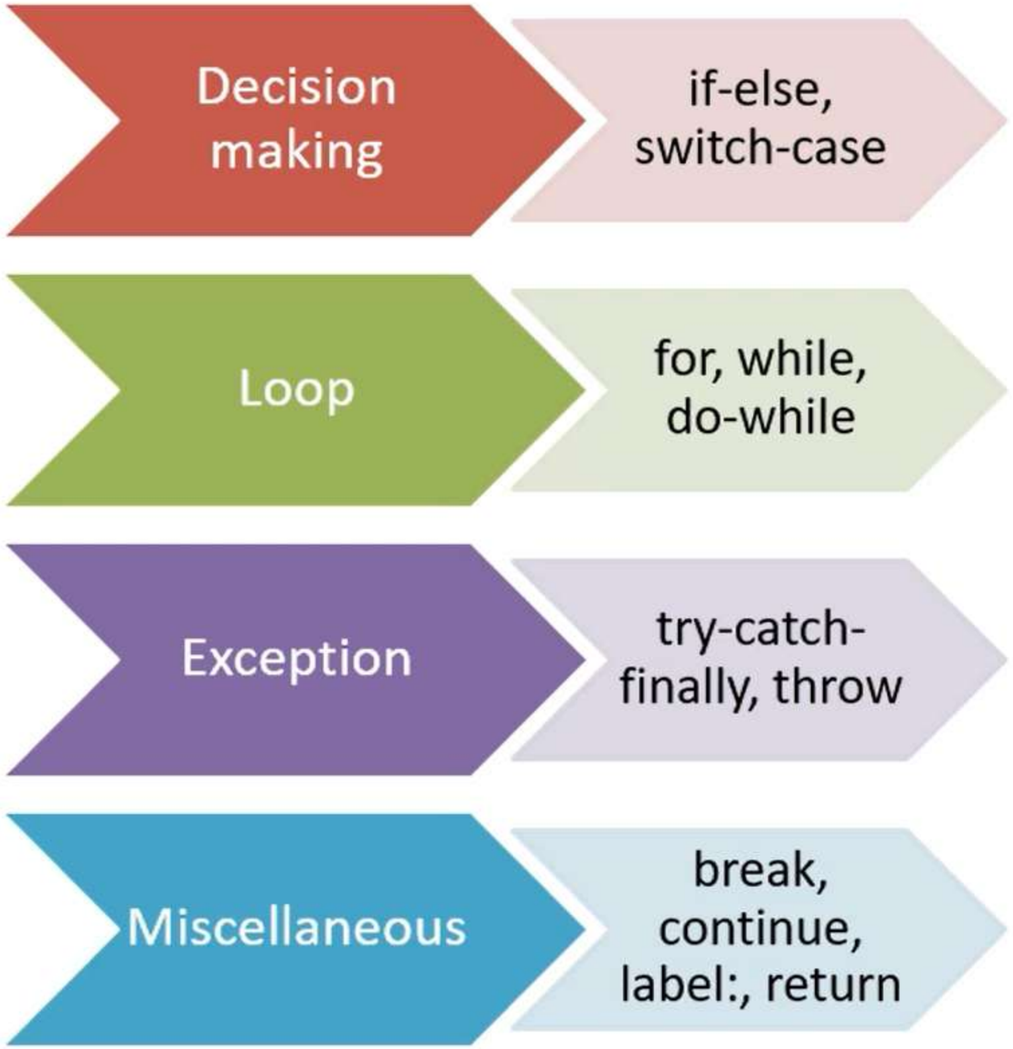


Control Structures - Overview



Control Structures

TEKNOTURF

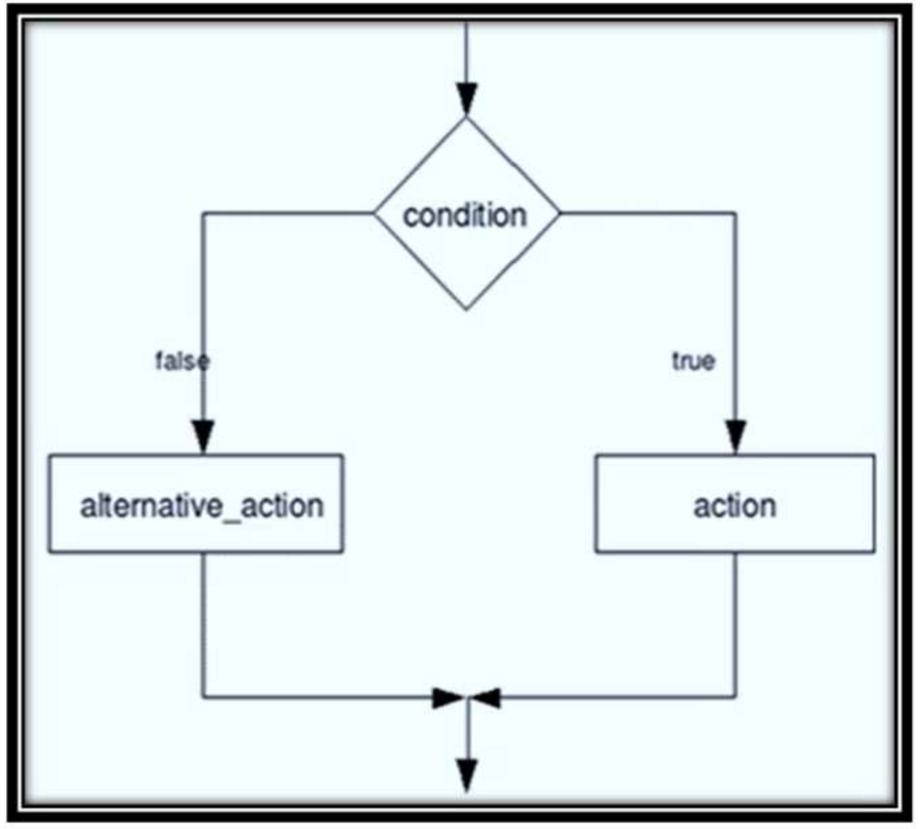


Decision Making – if else

TEKNOTURF

SYNTAX

```
if(condition) {
      //true part
}
else  {
      //false part
 }
```



Example

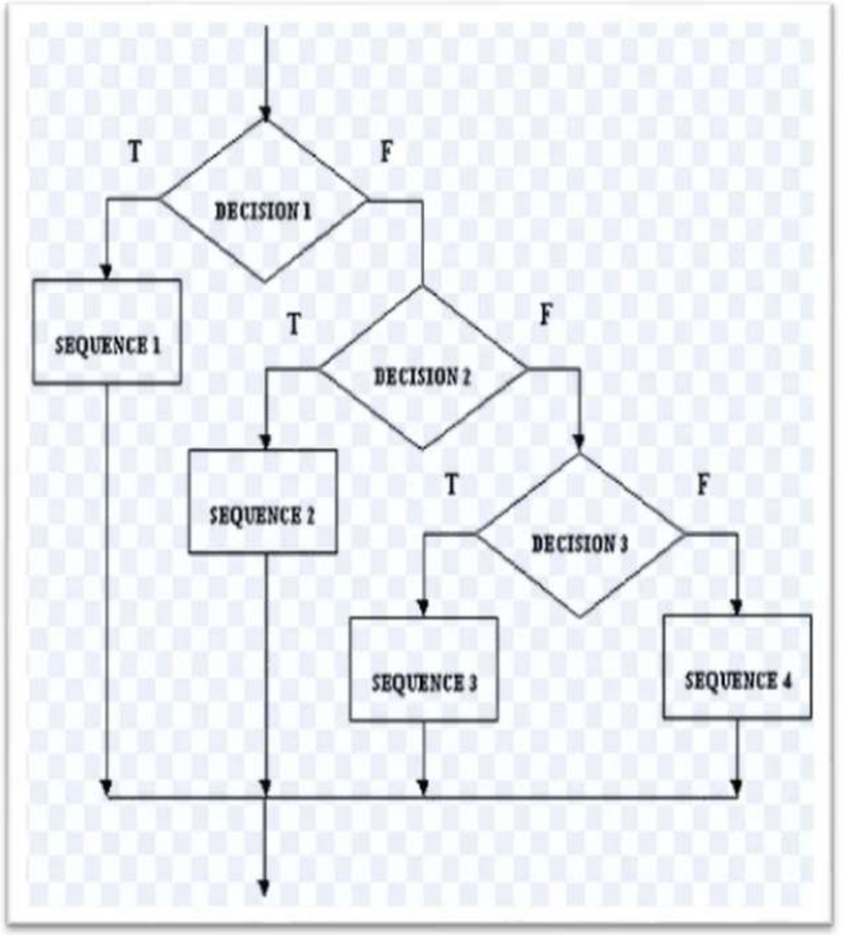
```
if(num1 > num2)
                System.out.println("num1 is greater");
else
                System.out.println("num2 is greater");
```

Decision Making – Nested If

TEKNOTURF

SYNTAX

```
if(condition1){
      if(condition2) {
            // true part of both conditions
      }
      else {
            // false part of condition2
       }
}
else{
      // false part of condition1
}
```



Decision Making – Nested If

TEKNOTURF

- **Example**

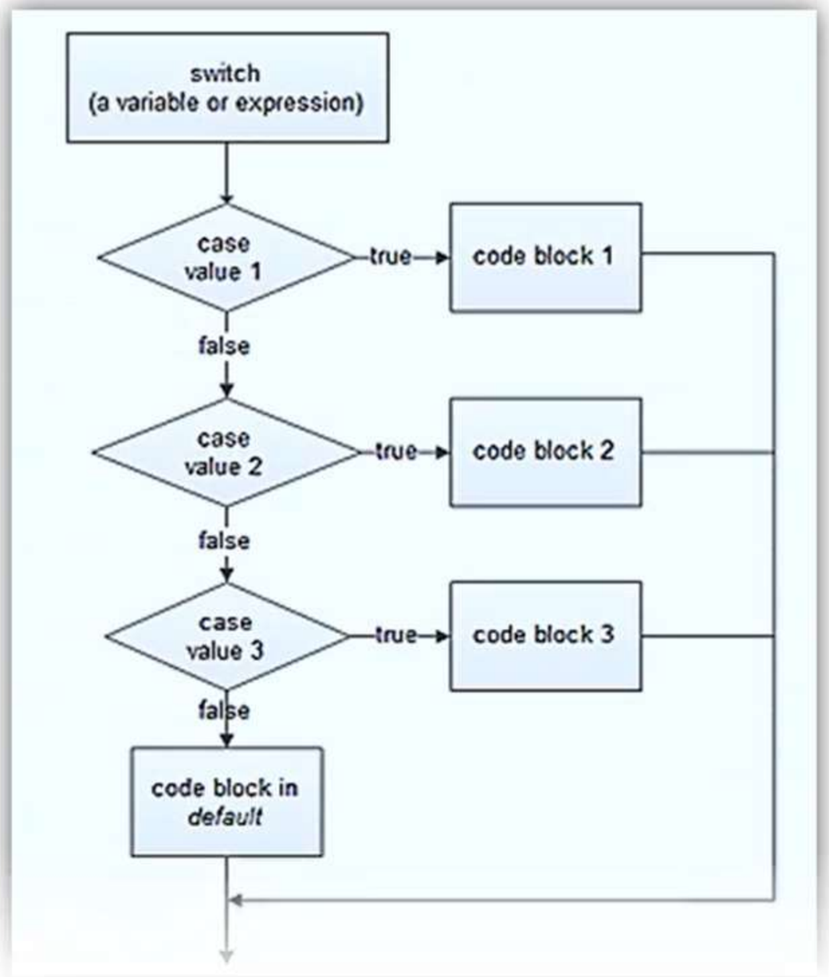
```
if (age <= 19 ) {
        if (age >= 13) {
                System.out.println( "You are in your teen age");
        }
        else {
                System.out.println( "You are a Child");
        }
}
else {
        System.out.println(" You are an adult ");
 }
```

Decision Making – switch - case

TEKNOTURF

SYNTAX

```
switch(parameter) {
        case 1:
            //block executed when
           // the int parameter value is 1
           break;
        case 3:
            //block executed when
           //the int parameter value is 3
           break;
        default:
            // block executed when
           // the int parameter value does not match  any specific value
}
```



Decision Making – switch - case

TEKNOTURF

Example

```
switch(grade) {
        case 'A' :
           System.out.println("Excellent!");   break;
        case 'B' :
          System.out.println("Well done");   break;
        case 'D' :
           System.out.println("You passed");  break;
        case 'F' :
           System.out.println("Better try again");  break;
        default :
           System.out.println("Invalid grade");
}
System.out.println("Your grade is " + grade);
```

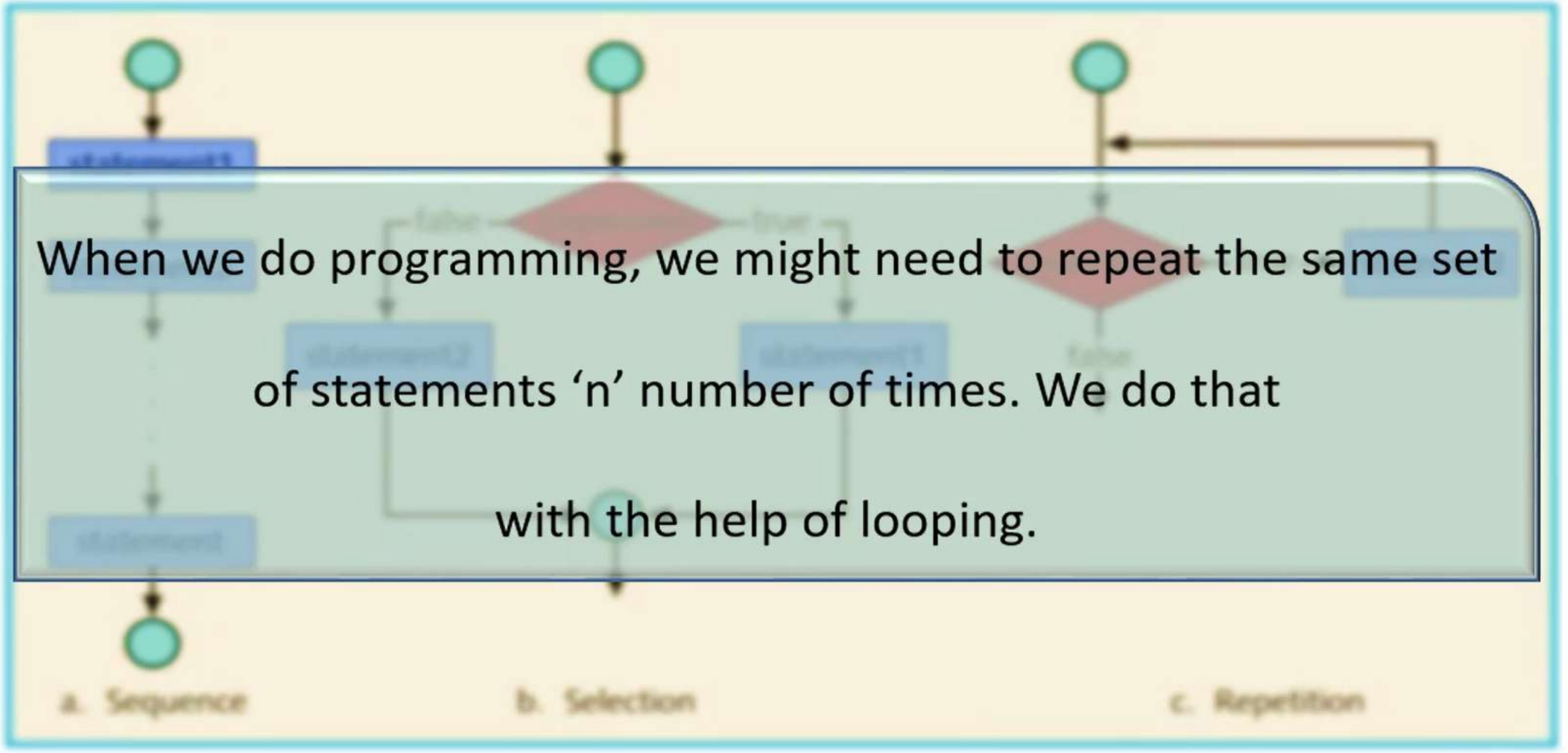
From Java 7, a string literal / constant can be used to control a switch statement

Overview

Sure. You have given a wrong PIN thrice. It will allow three times. If you exceed , the card will get blocked

Excuse me. My ATM Card got blocked. May I know the reason for that ?

Overview



When we do programming, we might need to repeat the same set of statements ‘n’ number of times. We do that with the help of looping.

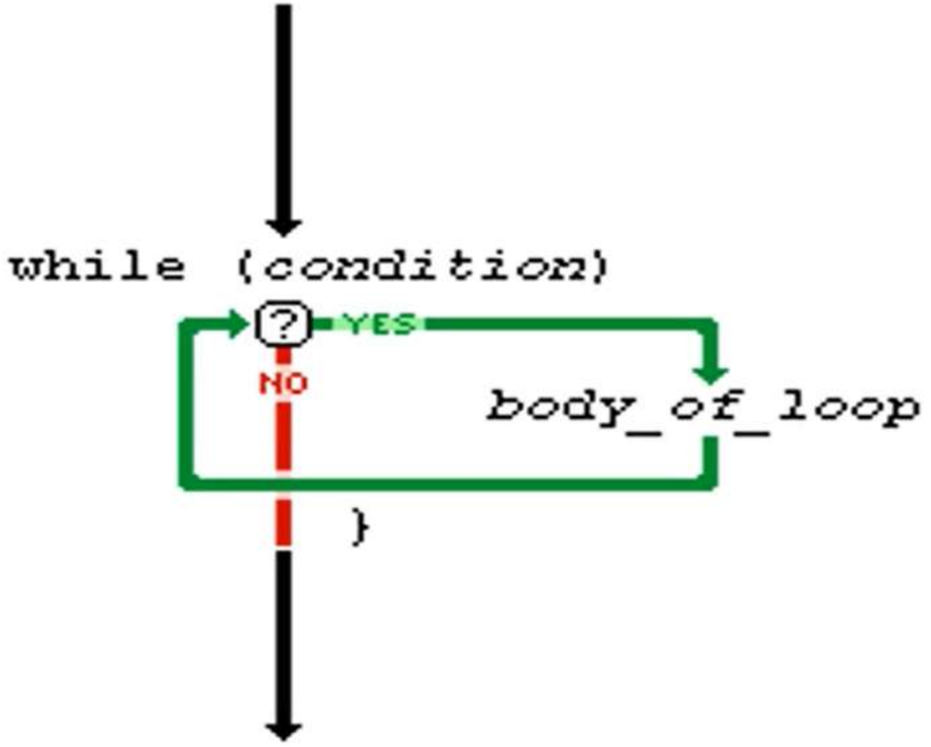
While Loop

TEKNOTURF

Example

```
int count = 1;
while (count < 11) {
    System.out.println(count);
     count++;
}
```

SYNTAX



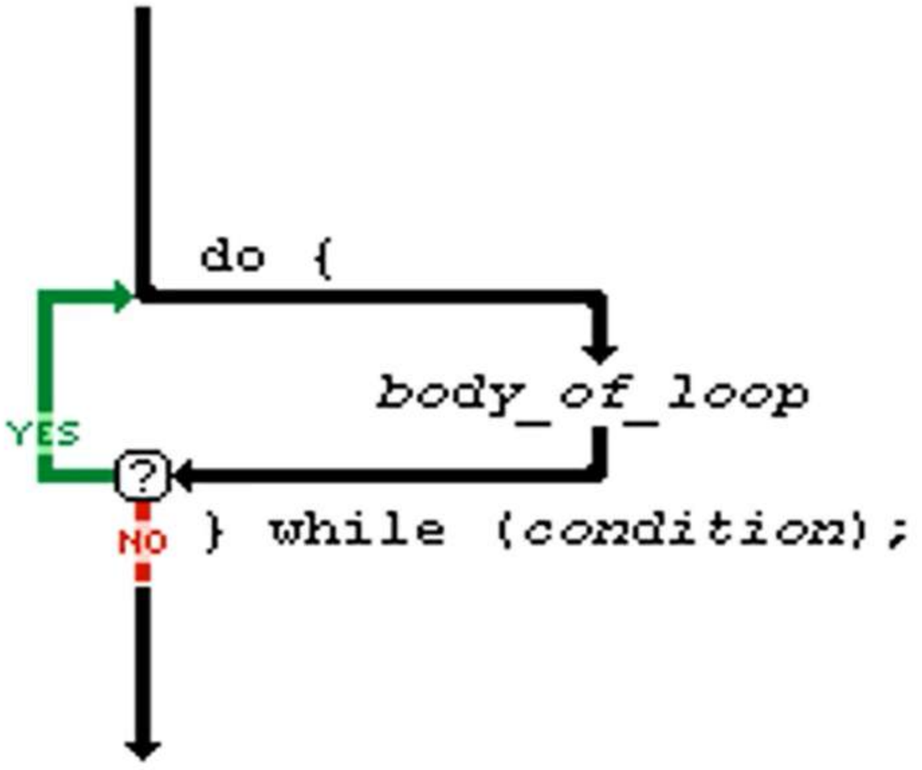
Do-While Loop

TEKNOTURF

Example

```
int count = 1;
    do {
        System.out.println(count);
       count++;
     } while (count < 11);
```

SYNTAX



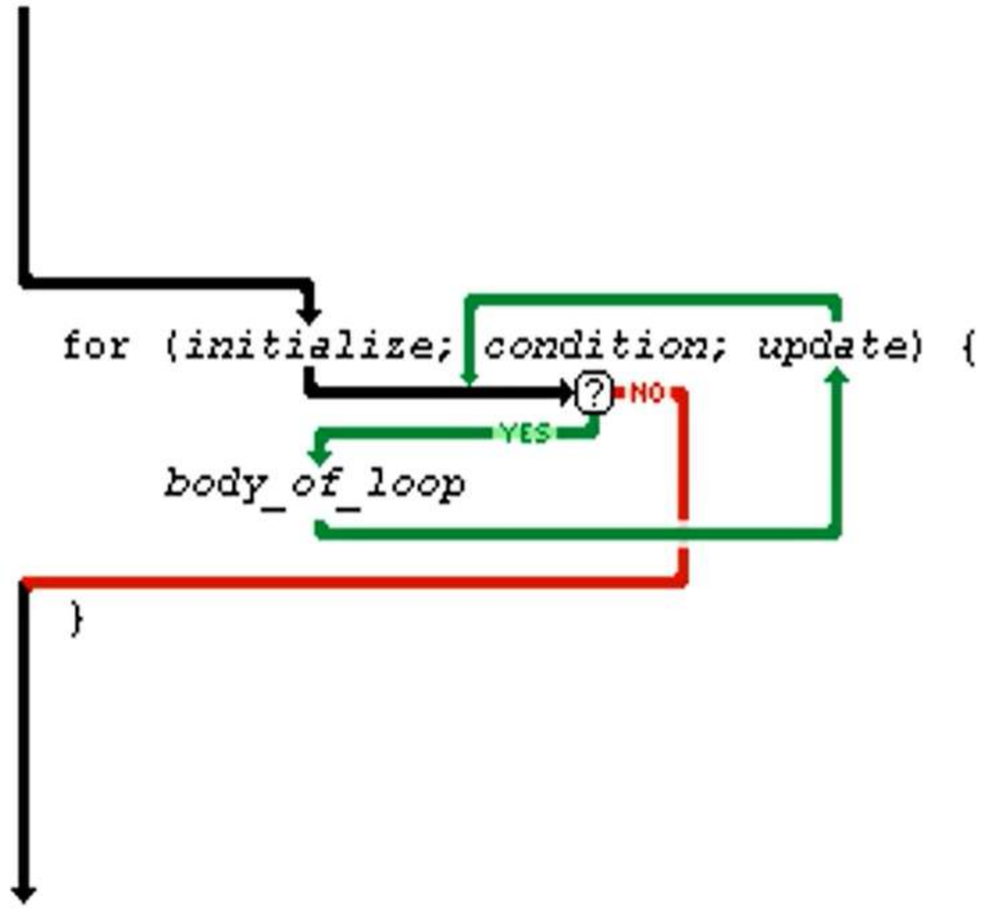
For Loop

TEKNOTURF

Example

```
for(int i=1; i<11; i++) {
        System.out.println("Count is: " + i);
}
```

SYNTAX



break statement

TEKNOTURF

break statement

- Used to terminate a loop as well as switch - case block
- When executed, the control flow will be transferred to the statement that follows after the end of the loop
- Example

```
int a=0,b=1,c=0,cnt=2;
System.out.print(a+", "+b);
while(cnt<20)
{
    c=a+b;
    System.out.print(", "+c);
    if(c%11==0)
        break;
    cnt++;
    a=b;
    b=c;
}
```

Output :

0,1,1,2,3,5,8,13,21,34,55

continue Statement

TEKNOTURF

continue statement

- Used to bypass the execution of rest of the statements in the current iteration of a loop
- It skips to the end of the inner loop and continues the loop
- Example

```
int a=0,b=1,c=0,cnt=2;
System.out.print(a+", "+b);
while(cnt<15)
{
    c=a+b;
    cnt++;
    a=b;
    b=c;
    if(c%11==0)
        continue;
     System.out.println(", "+c);
}
```

Output :

0,1,1,2,3,5,8,13,21,34, 89,144,233,377

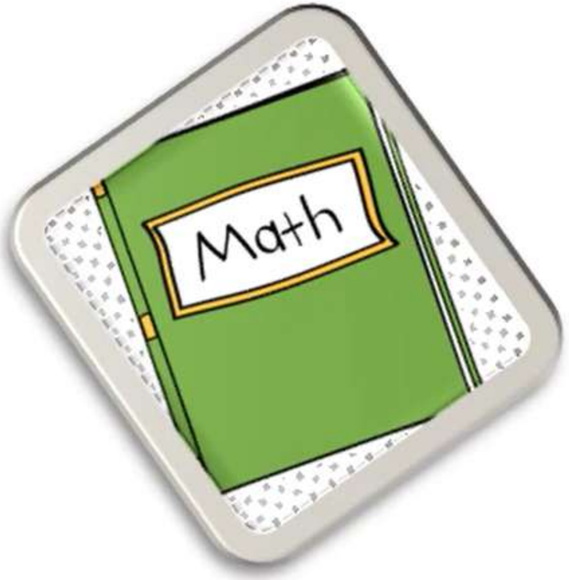
Labeled Statements

TEKNOTURF

Labels can be used with loop statements like for or while, in conjunction with break and continue statements.

A label statement must be placed just before the statement being labeled

Label consists of a valid identifier that ends with a colon (:)



Labeled Statements - Example

TEKNOTURF

Example

```
boolean isTrue = true;
outer:
  for(int i=0; i<5; i++) {
    while (isTrue) {
        System.out.println("Hello");
        break outer;
    } // end of inner while loop
    System.out.println("Outer loop."); // Won't print
  } // end of outer for loop
System.out.println("Good-Bye");
```

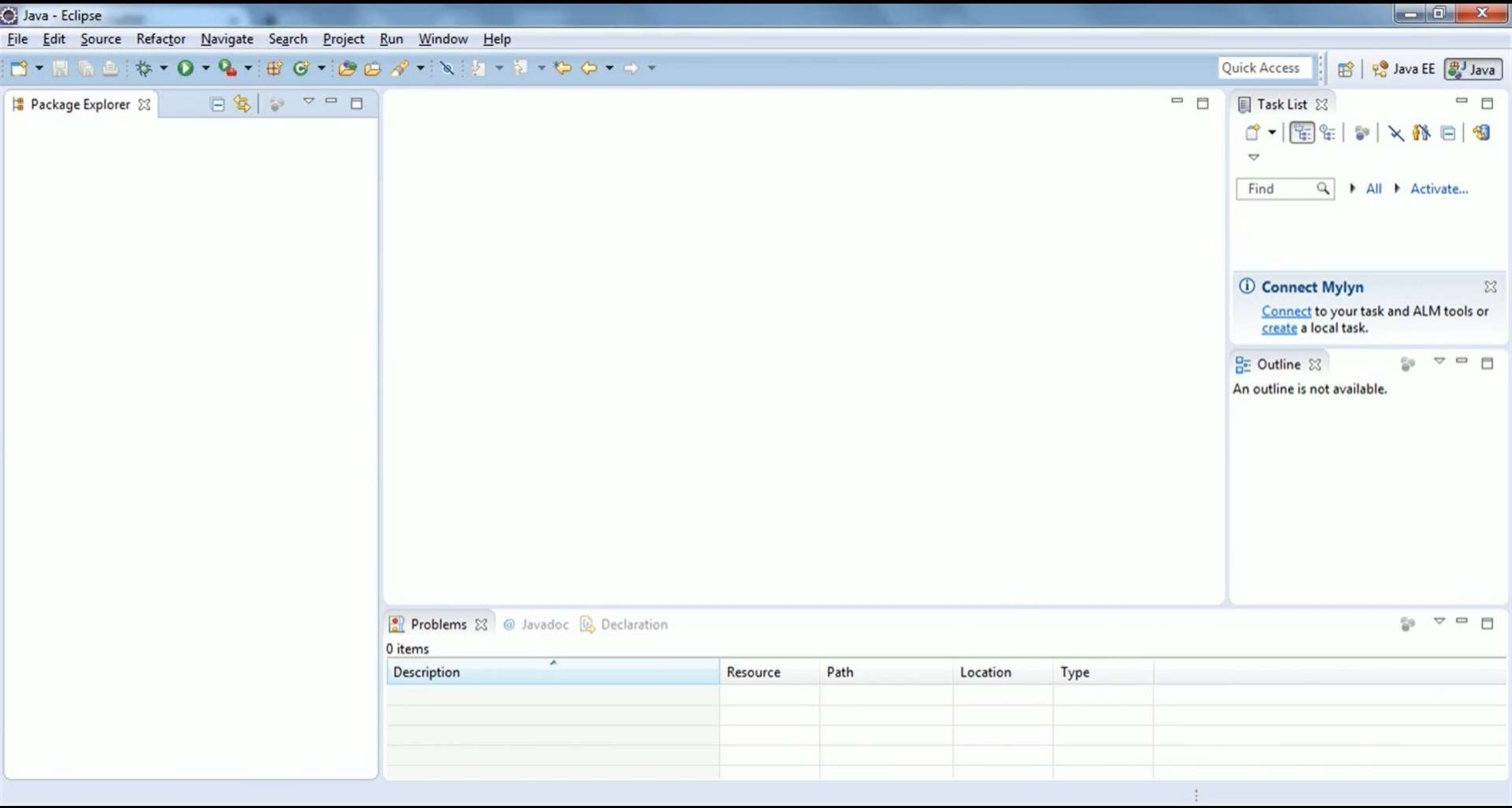
Running this code produces

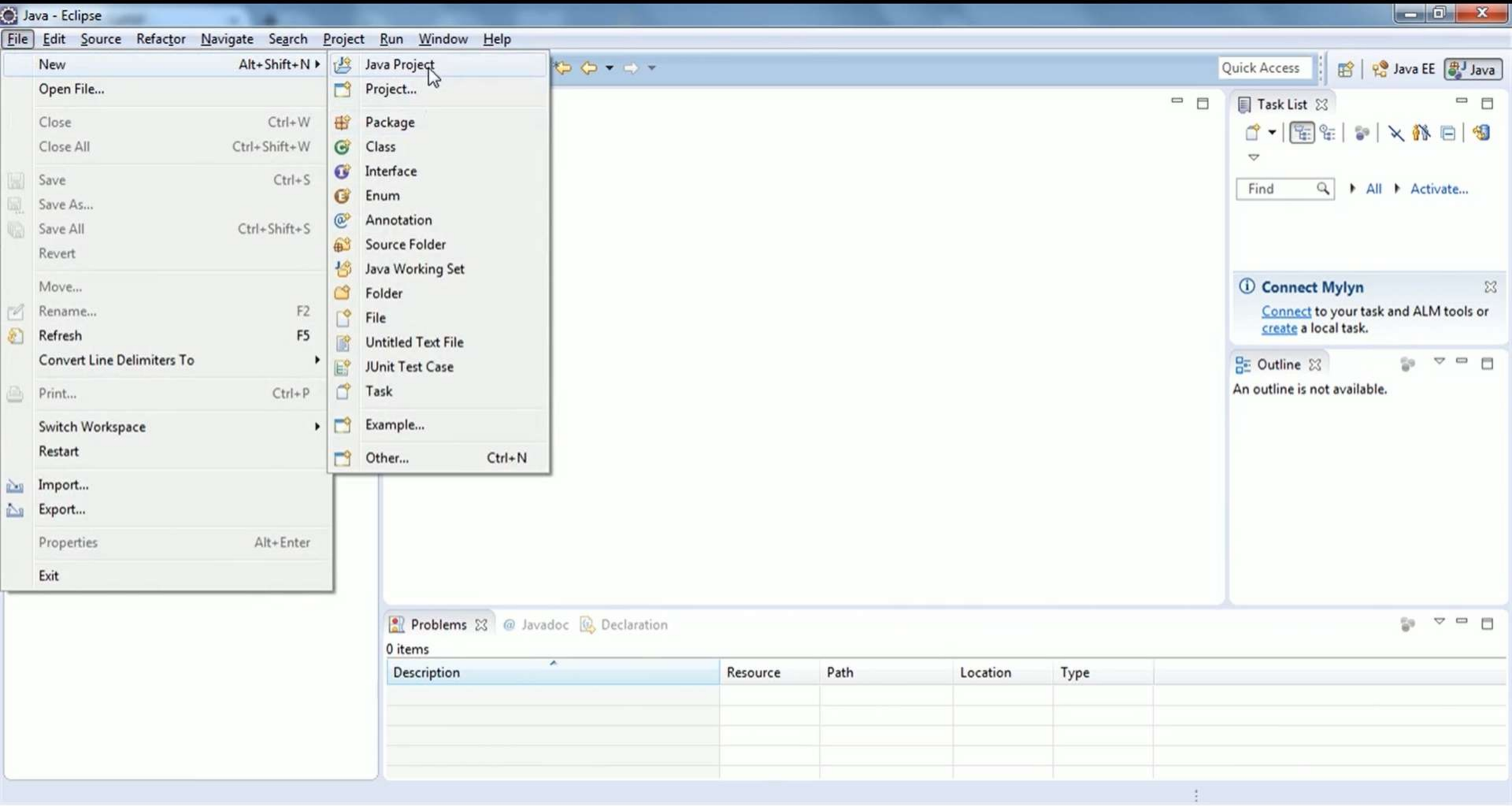
Hello

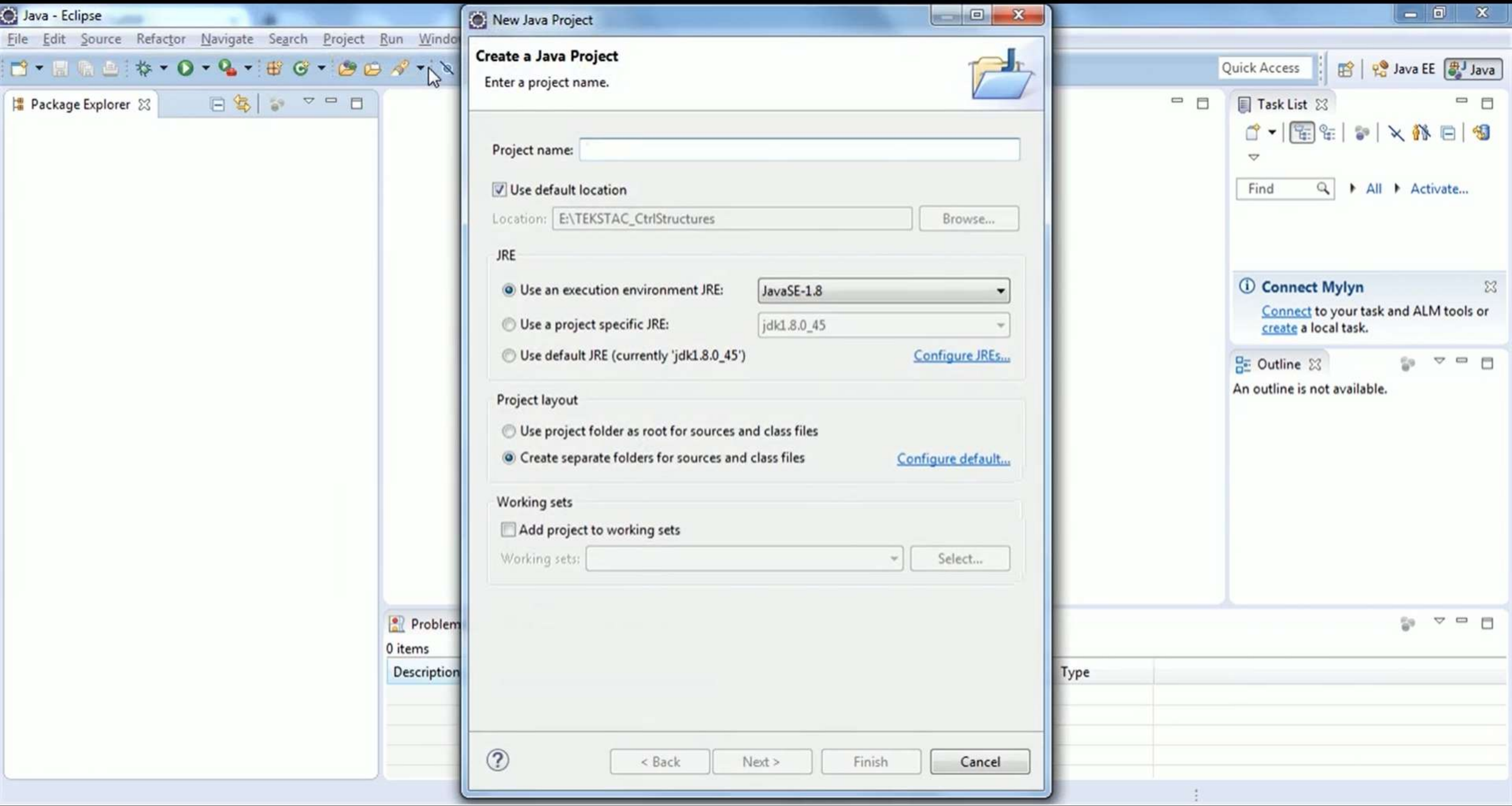
Good-Bye

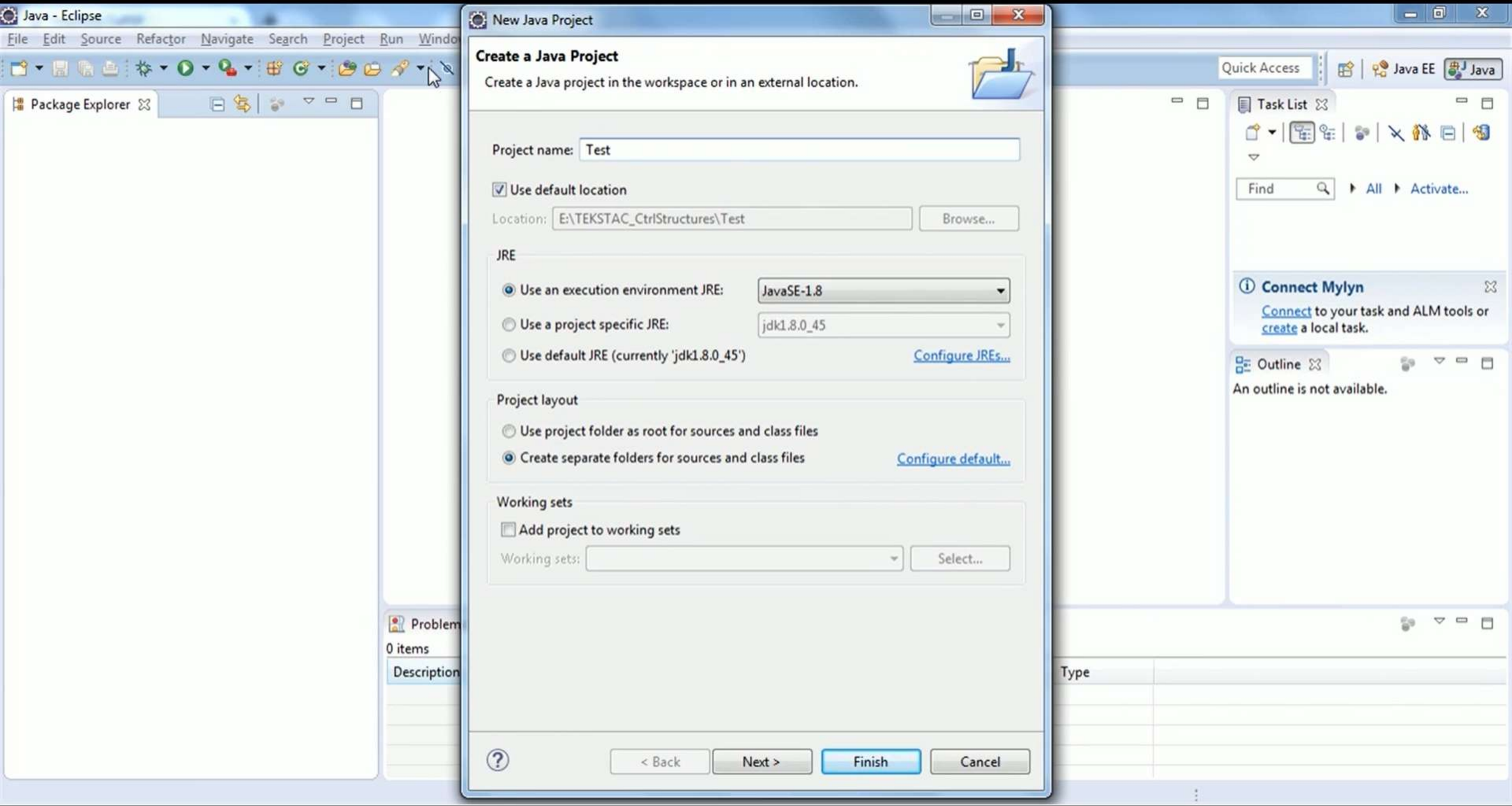
Summary

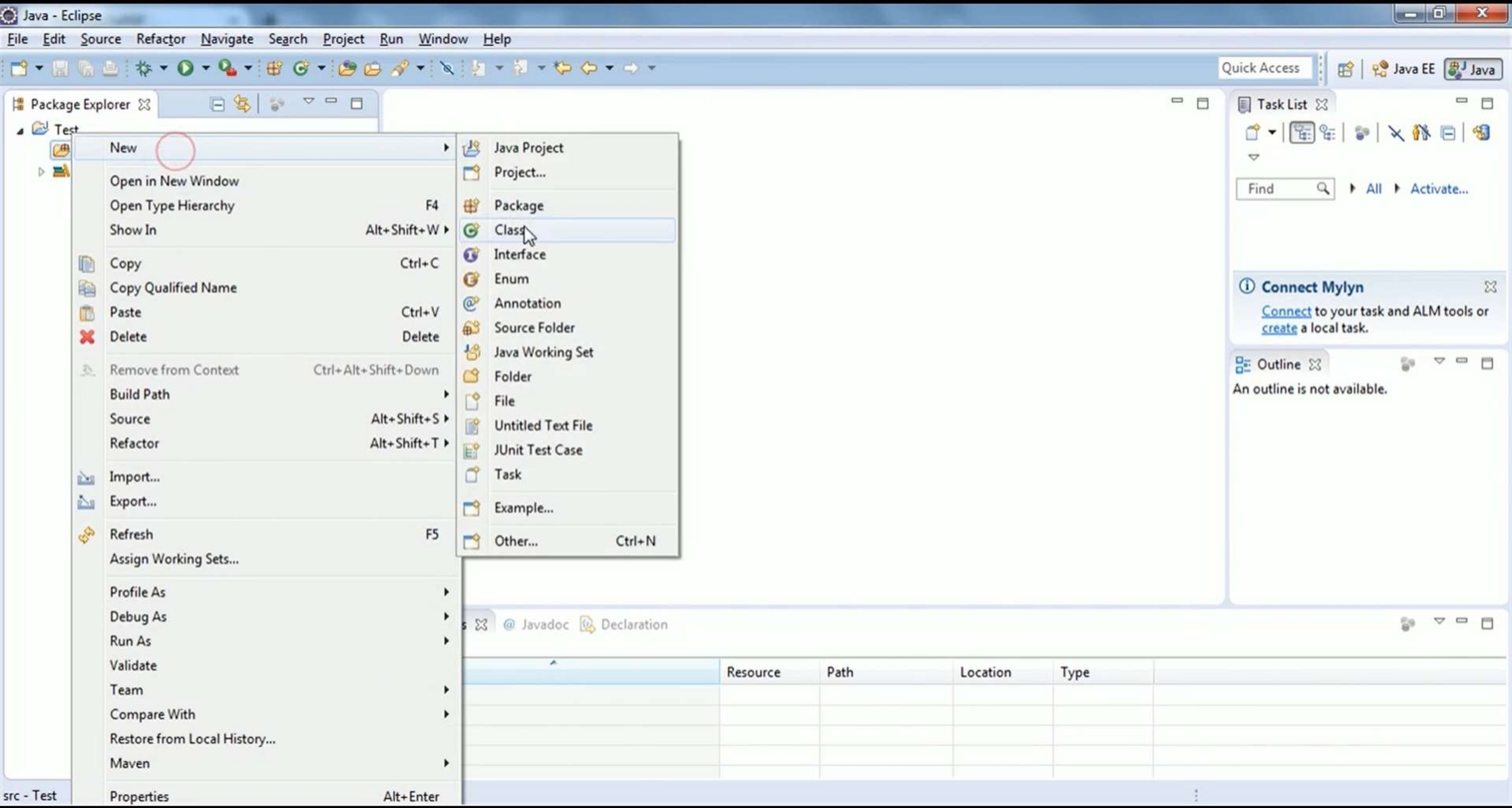
- Selection Statement
- Iterative Statement

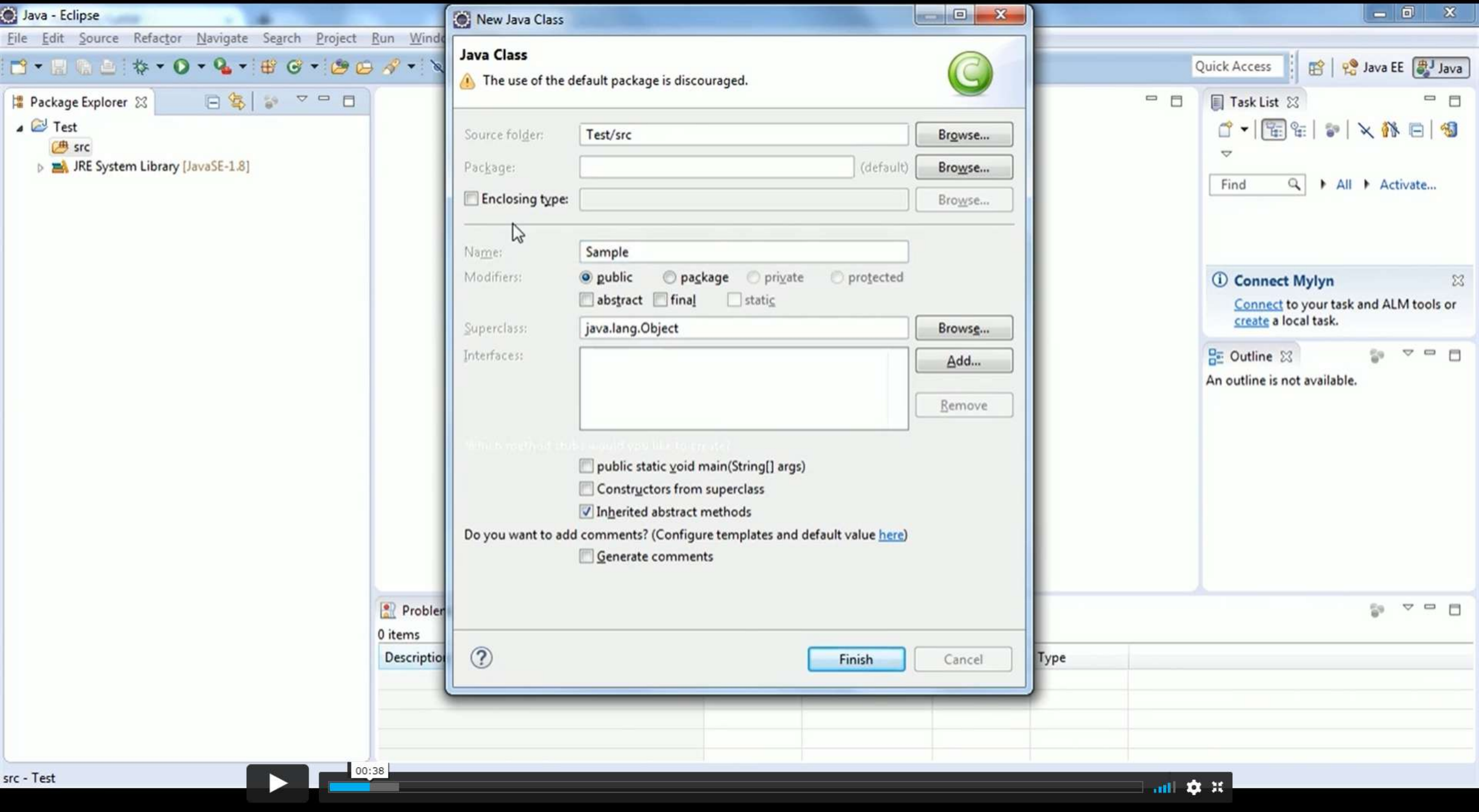

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

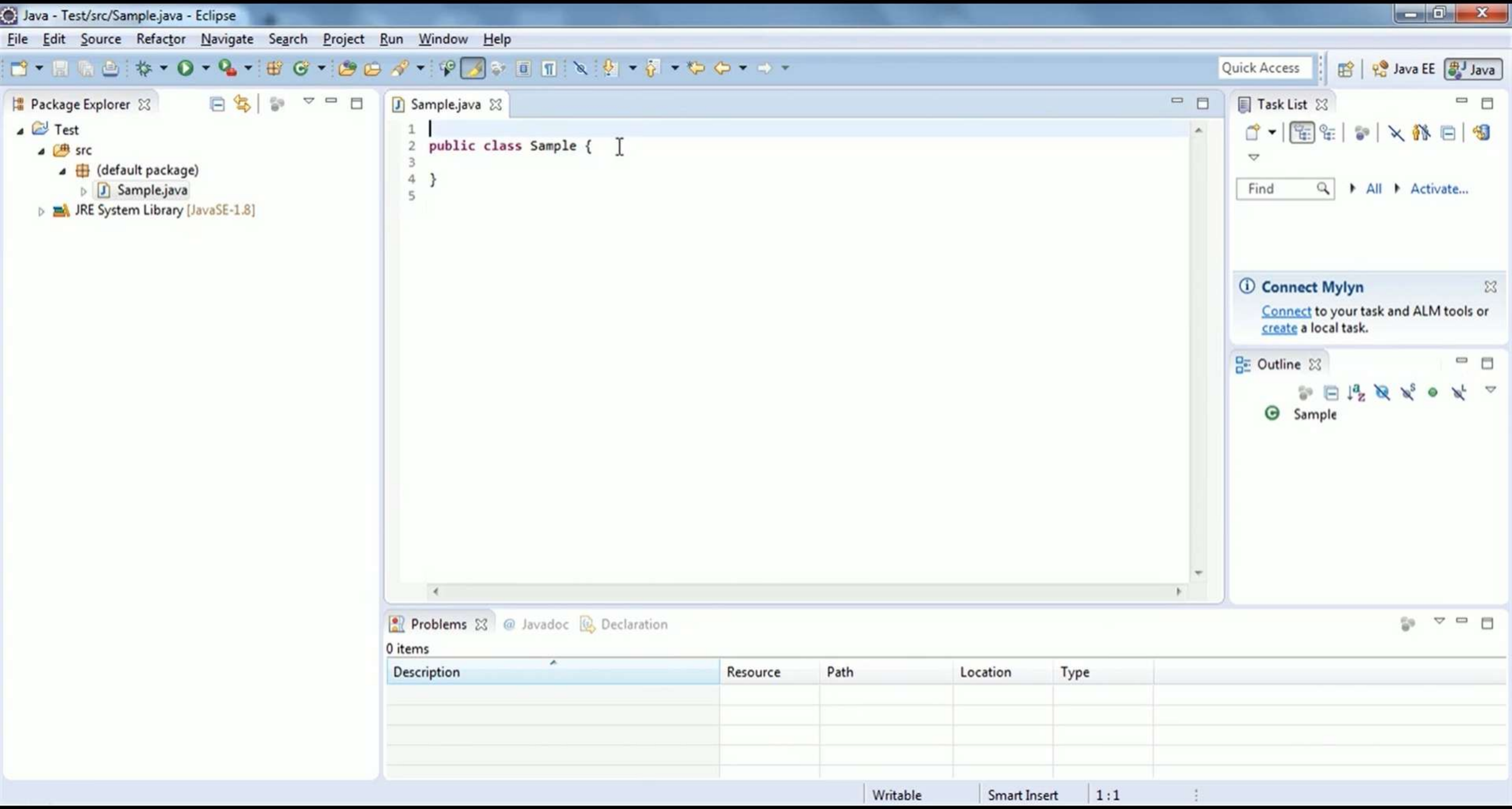

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
Exit
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... Ctrl+N
Quick Access
Java EE
Java
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

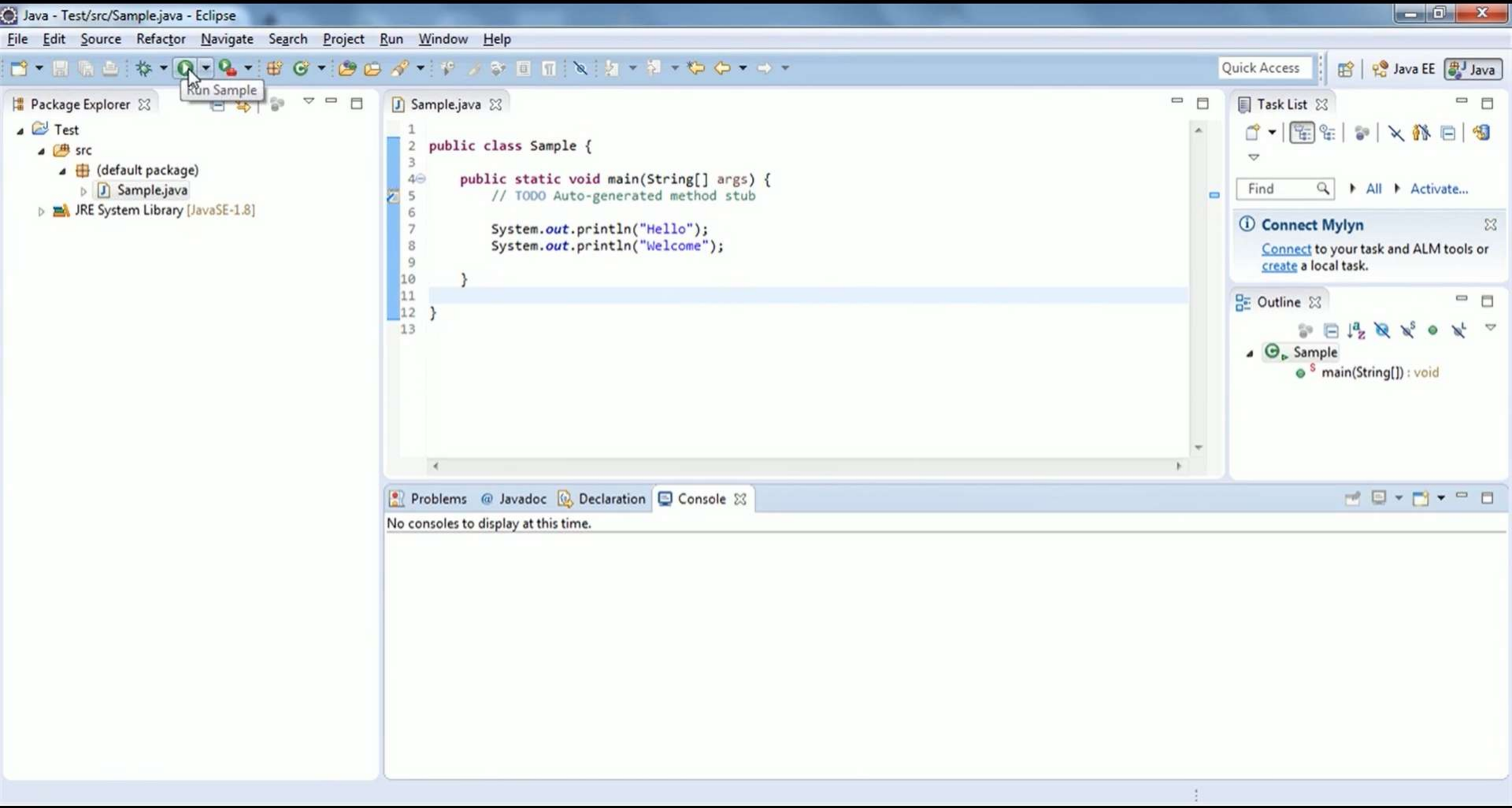

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Windo
Quick Access
Java EE
Java
Package Explorer
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problem
0 items
Description
Type
New Java Project
Create a Java Project
Enter a project name.
Project name:
Use default location
Location: E:\TEKSTAC_CtrlStructures
Browse...
JRE
Use an execution environment JRE: JavaSE-1.8
Use a project specific JRE: jdk1.8.0_45
Use default JRE (currently 'jdk1.8.0_45')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

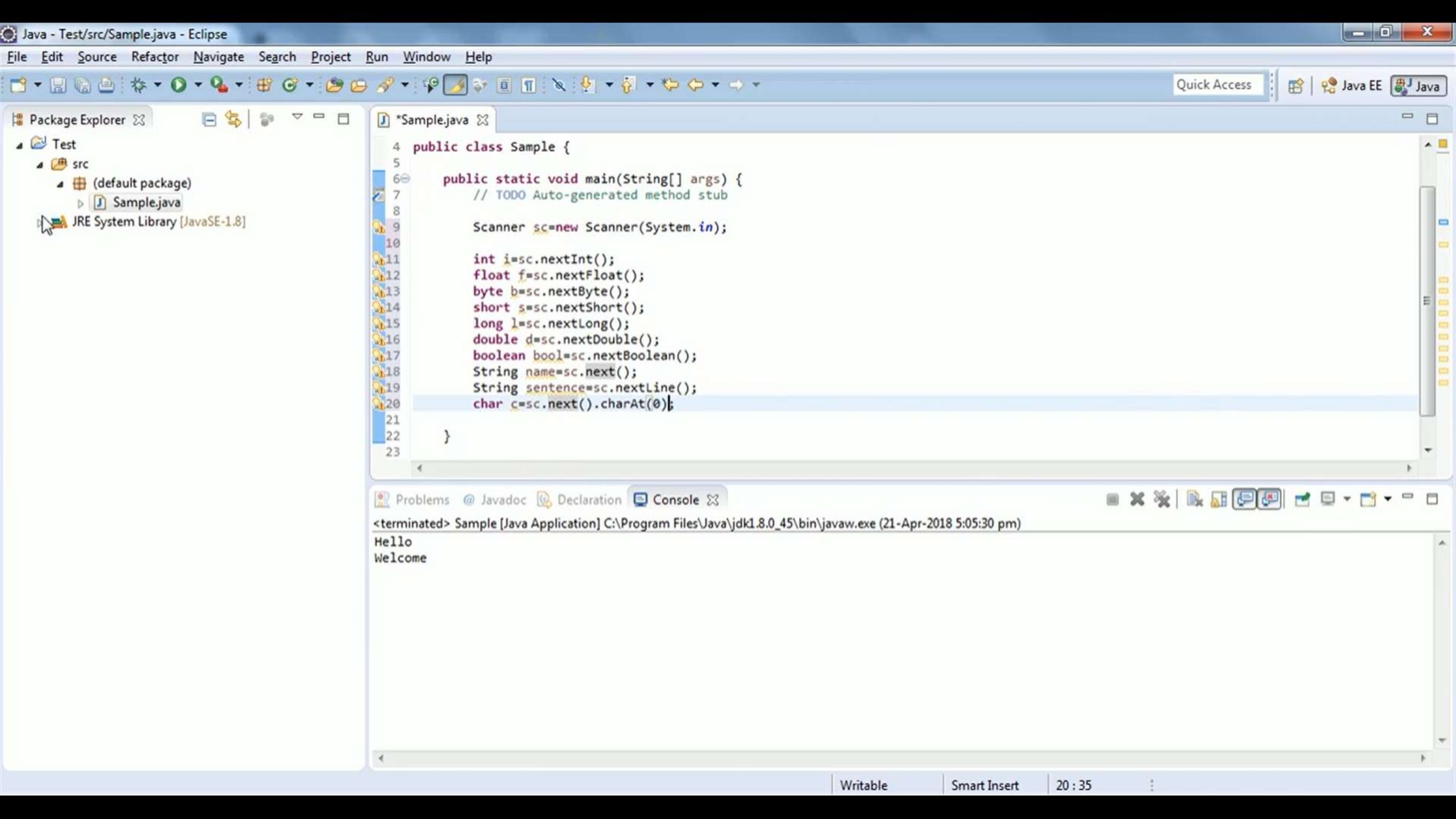

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Windo
Quick Access
Java EE
Java
Package Explorer
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problem
0 items
Description
Type
New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: Test
Use default location
Location: E:\TEKSTAC_CtrlStructures\Test
Browse...
JRE
Use an execution environment JRE: JavaSE-1.8
Use a project specific JRE: jdk1.8.0_45
Use default JRE (currently 'jdk1.8.0_45')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

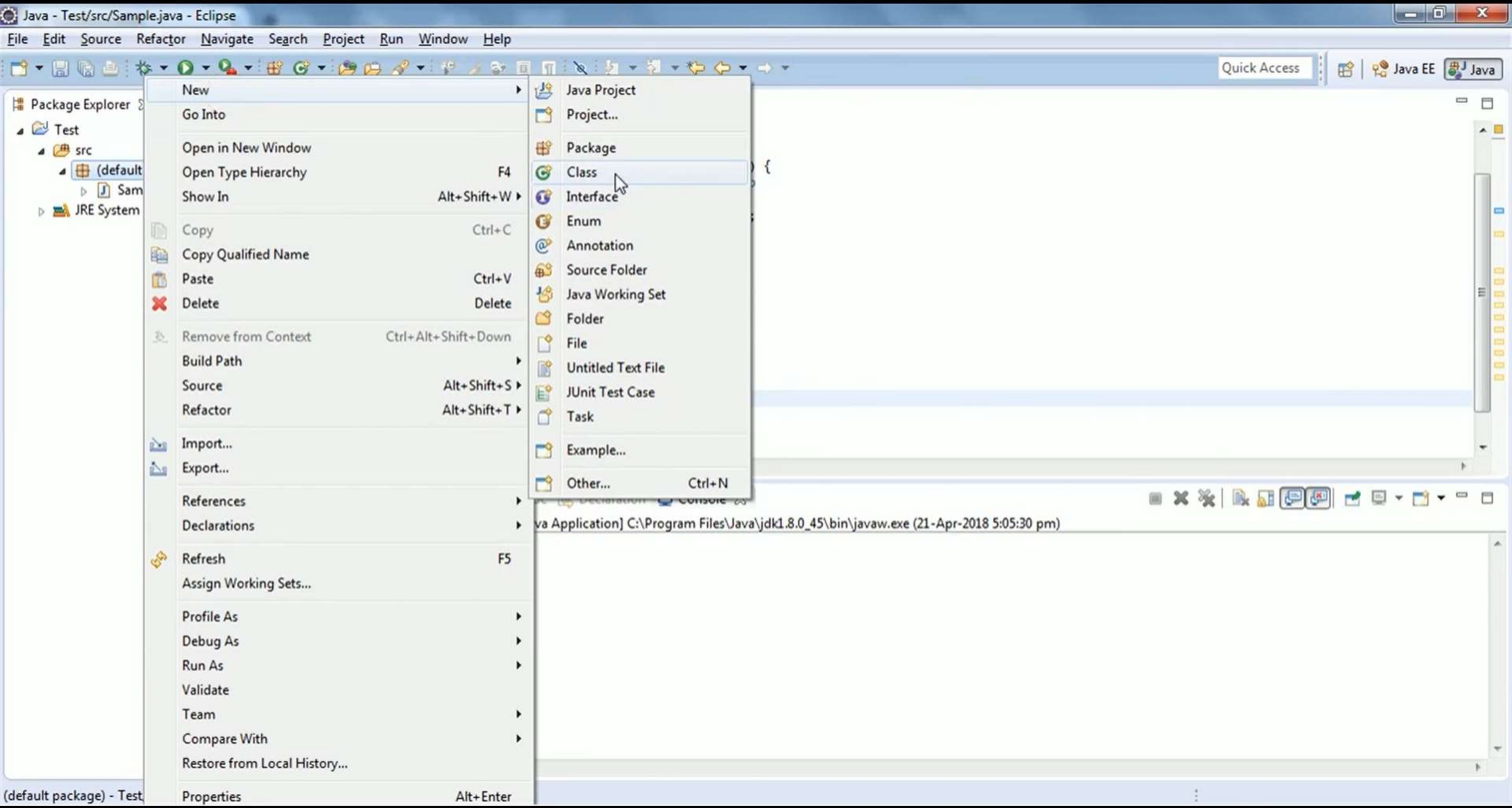

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
New
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Assign Working Sets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Restore from Local History...
Maven
Properties Alt+Enter
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... Ctrl+N
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Javadoc
Declaration
Resource
Path
Location
Type
src - Test

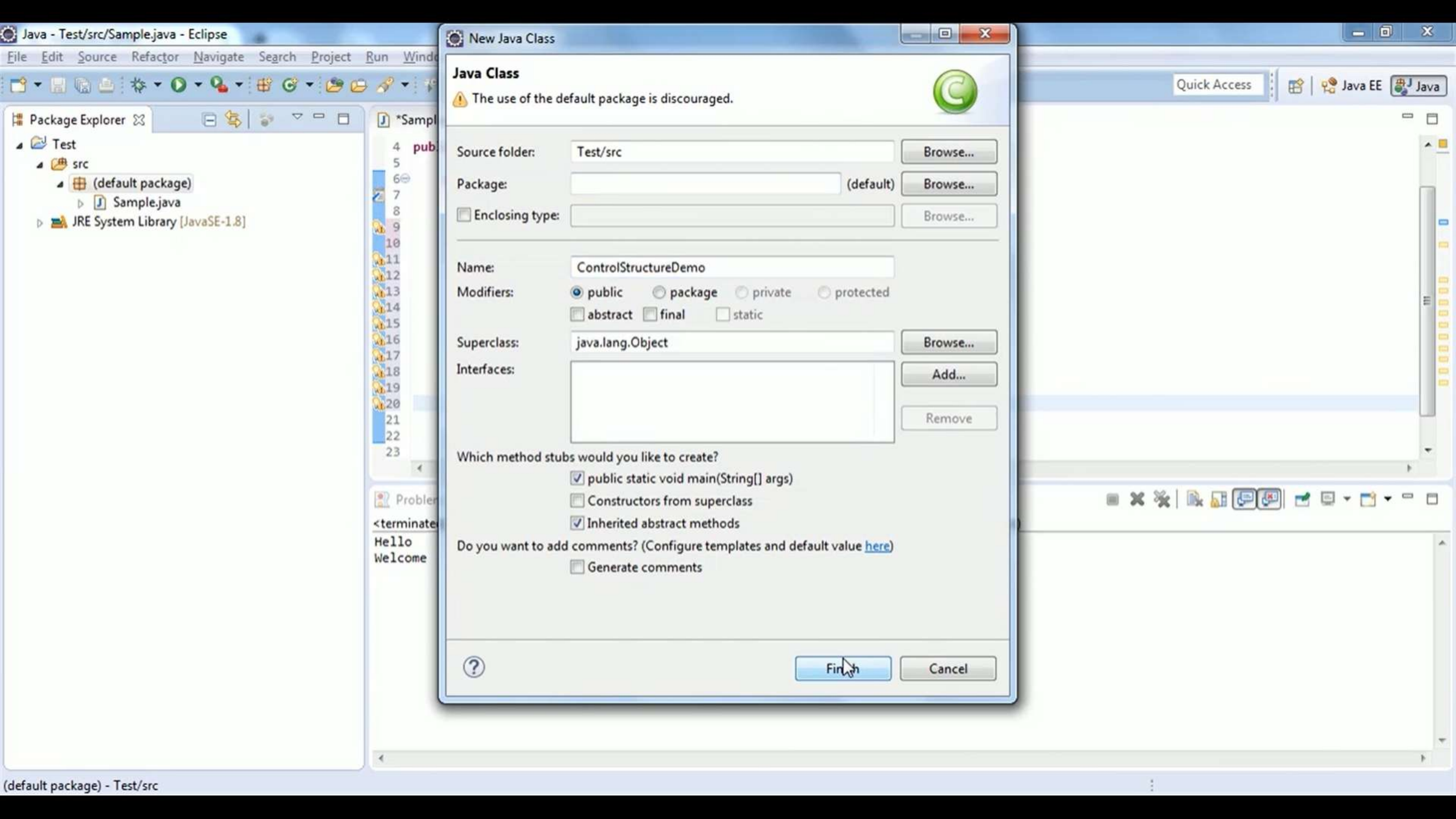

Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Windo
Quick Access
Java EE
Java
Package Explorer
Test
src
JRE System Library [JavaSE-1.8]
New Java Class
Java Class
The use of the default package is discouraged.
Source folder:
Test/src
Browse...
Package:
(default)
Browse...
Enclosing type:
Browse...
Name:
Sample
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
0 items
Description
Type
src - Test
00:38

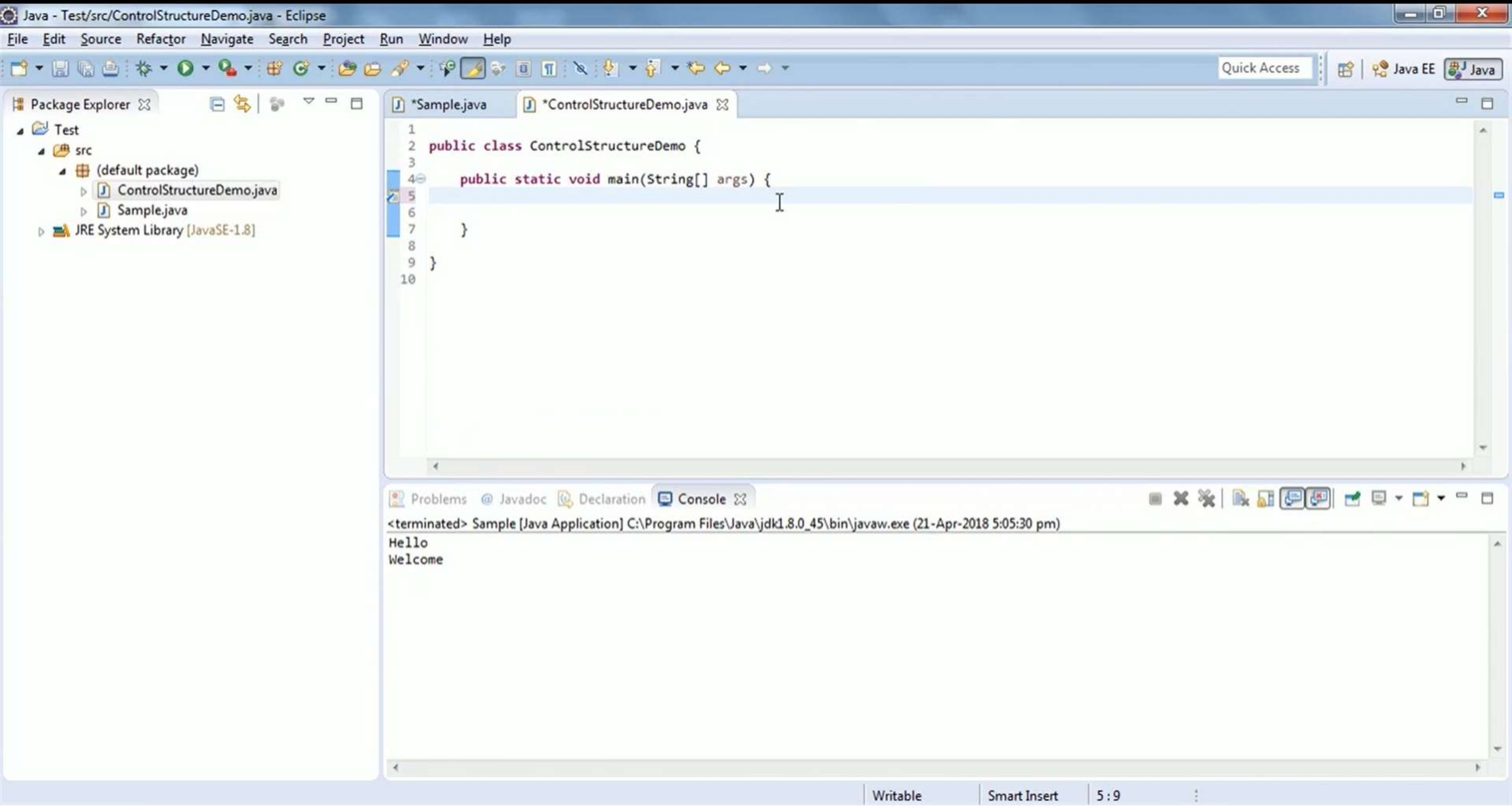

Java - Test/src/Sample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
1
2 public class Sample {
3
4 }
5
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
Sample
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type
Writable
Smart Insert
1 : 1

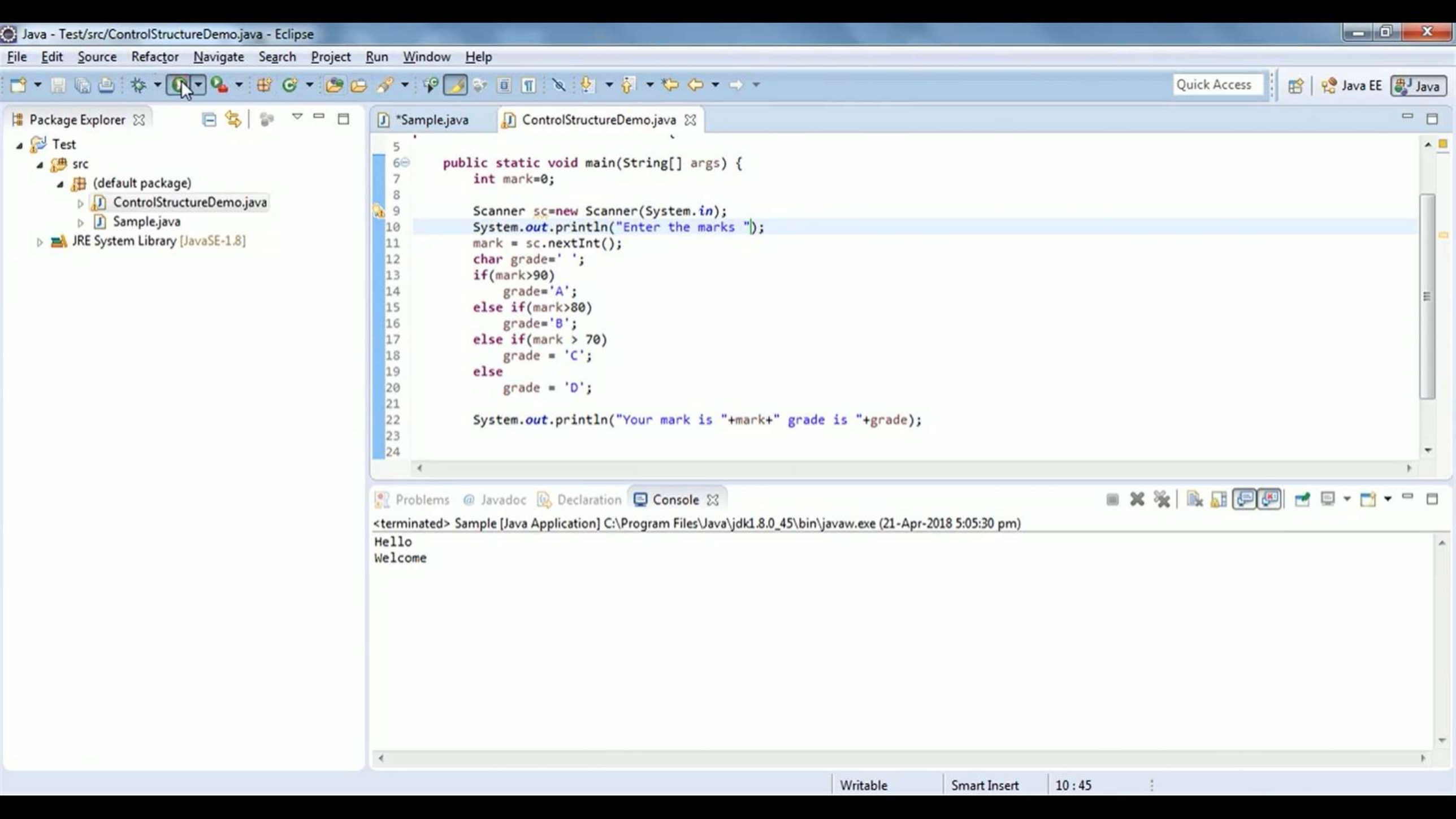

Java - Test/src/Sample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Run Sample
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
1
2 public class Sample {
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6
7 System.out.println("Hello");
8 System.out.println("Welcome");
9
10 }
11
12 }
13
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
Sample
main(String[]) : void
Problems
Javadoc
Declaration
Console
No consoles to display at this time.

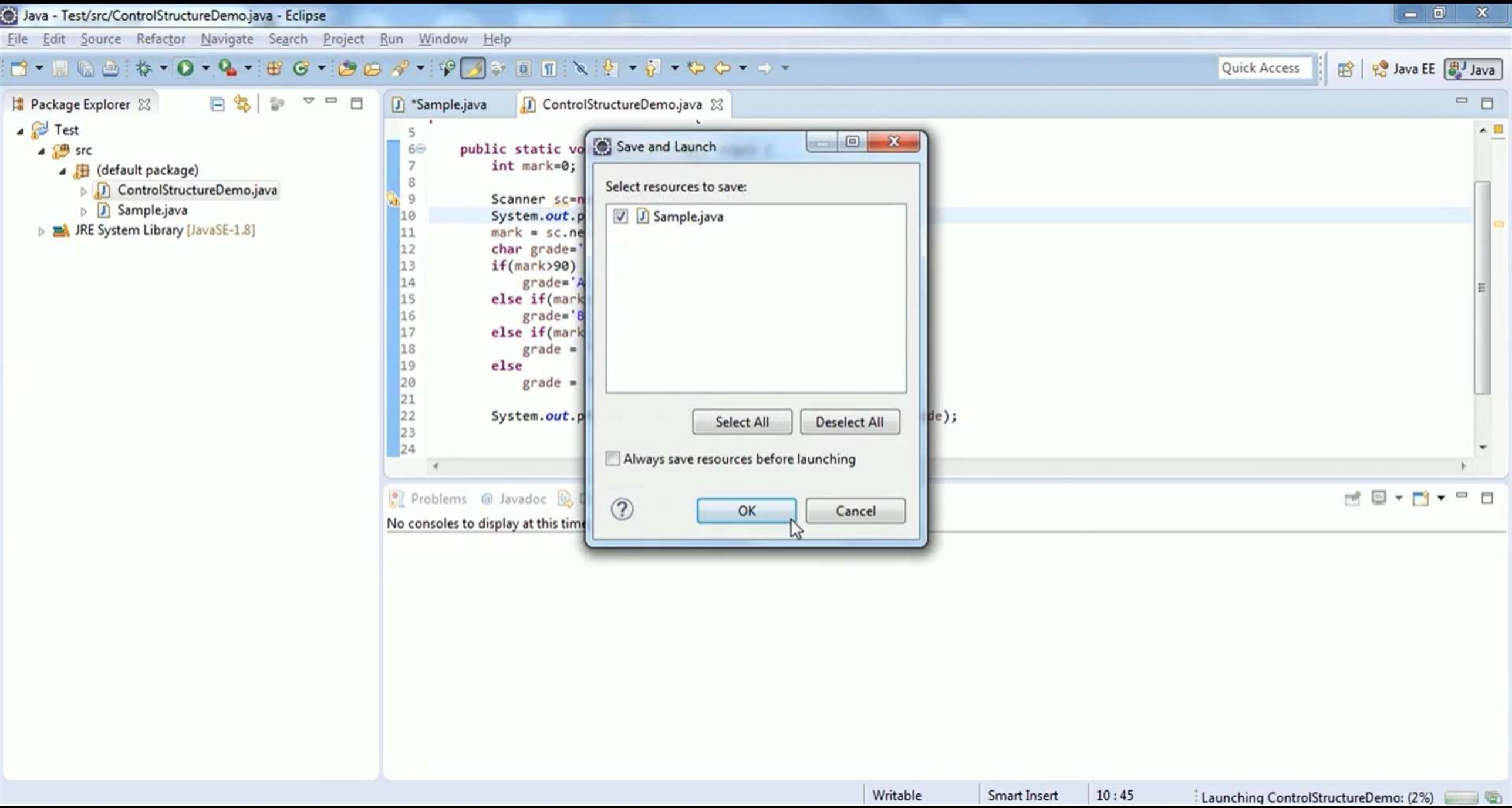

Java - Test/src/Sample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
Sample.java
JRE System Library [JavaSE-1.8]
*Sample.java
4 public class Sample {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8
9 Scanner sc=new Scanner(System.in);
10
11 int i=sc.nextInt();
12 float f=sc.nextFloat();
13 byte b=sc.nextByte();
14 short s=sc.nextShort();
15 long l=sc.nextLong();
16 double d=sc.nextDouble();
17 boolean bool=sc.nextBoolean();
18 String name=sc.next();
19 String sentence=sc.nextLine();
20 char c=sc.next().charAt(0);
21
22 }
23
Problems
Javadoc
Declaration
Console
<terminated> Sample [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:05:30 pm)
Hello
Welcome
Writable
Smart Insert
20 : 35

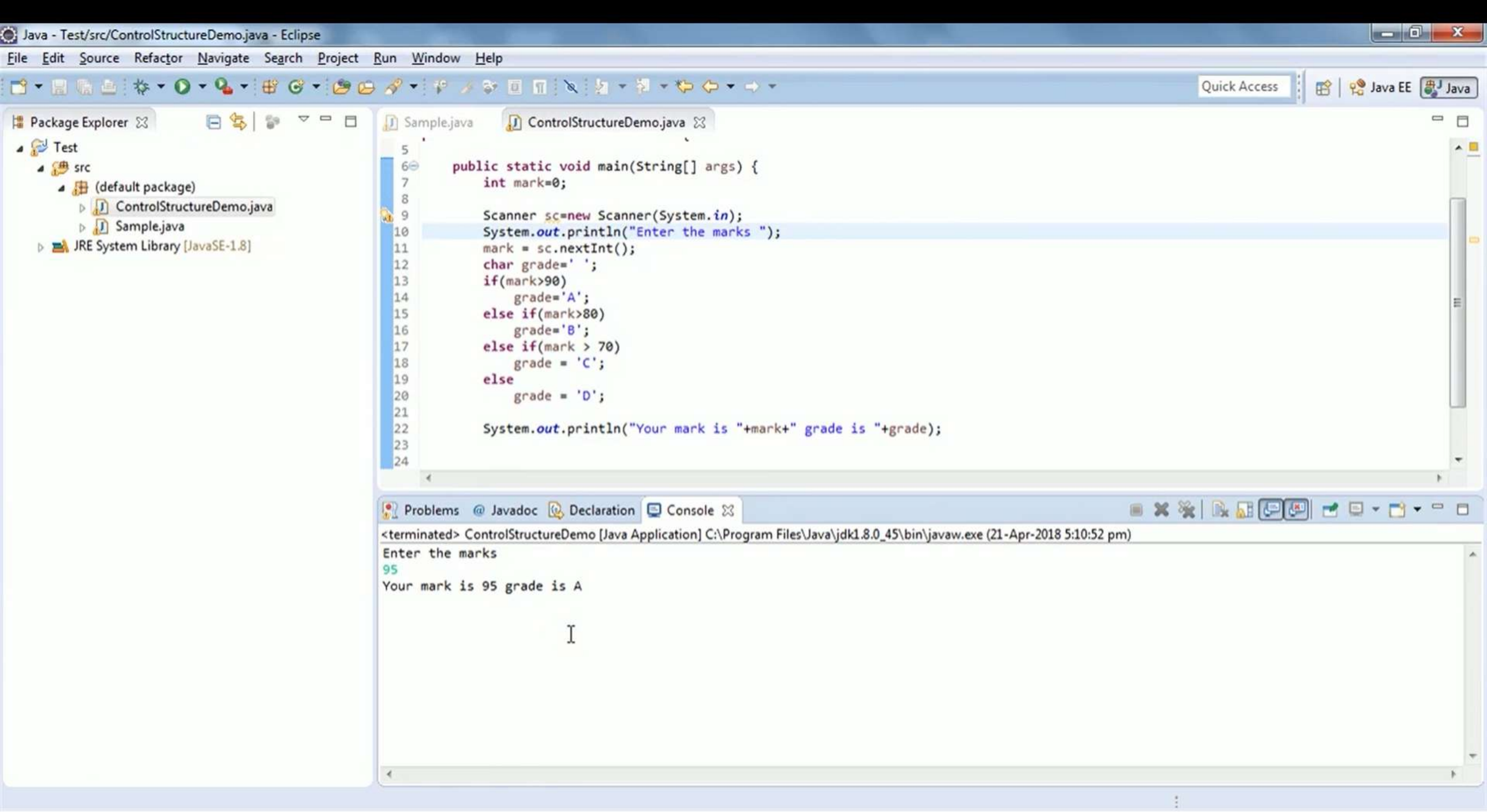

Java - Test/src/Sample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default
Sam
JRE System
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
References
Declarations
Refresh F5
Assign Working Sets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Restore from Local History...
Properties Alt+Enter
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... Ctrl+N
va Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:05:30 pm)
(default package) - Test

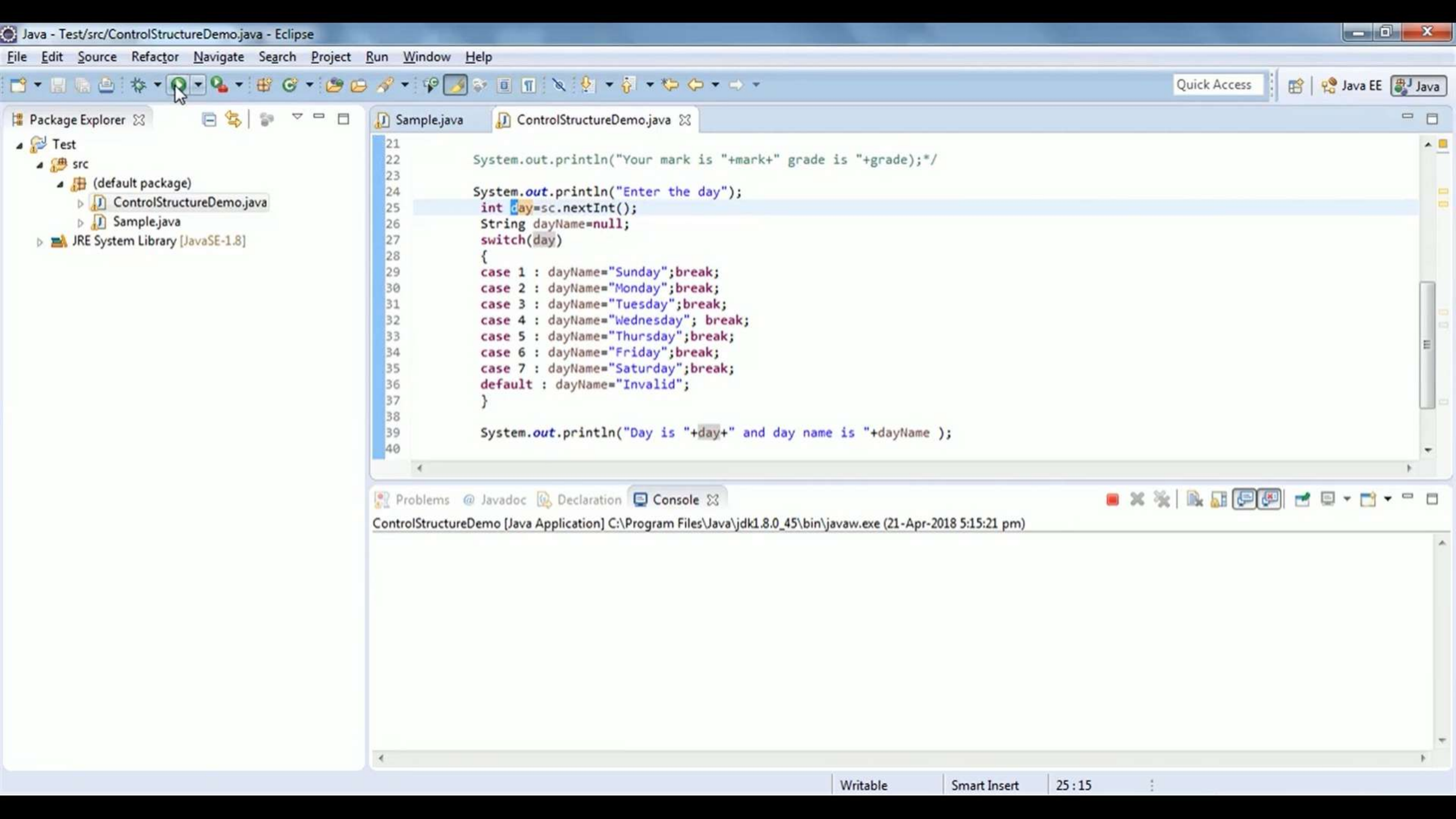

Java - Test/src/Sample.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Windo
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
Sample.java
JRE System Library [JavaSE-1.8]
*Sampl
4 pub
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Proble
<terminate
Hello
Welcome
(default package) - Test/src
New Java Class
Java Class
The use of the default package is discouraged.
Source folder:
Test/src
Browse...
Package:
(default)
Browse...
Enclosing type:
Browse...
Name:
ControlStructureDemo
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

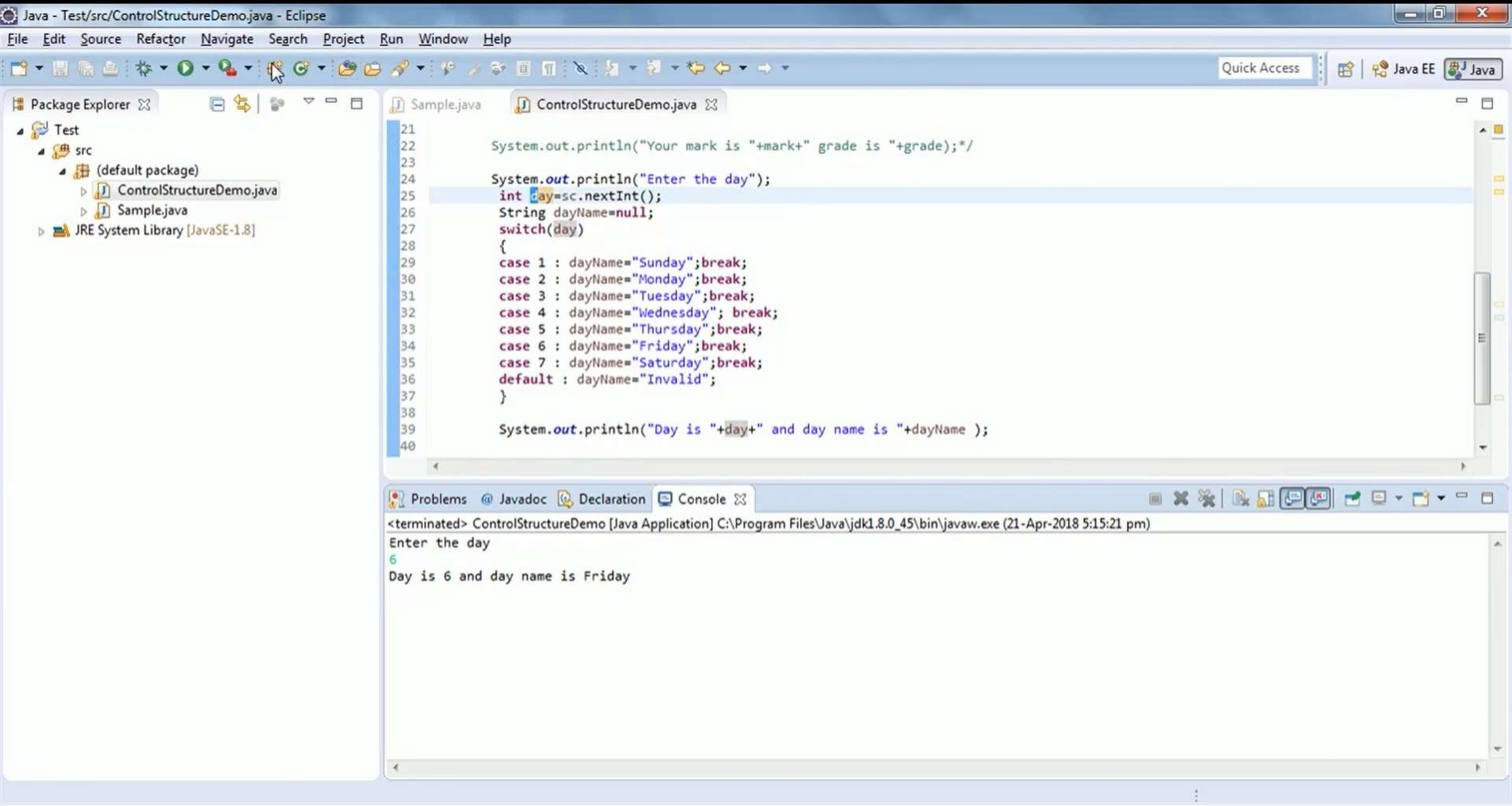

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
*Sample.java
*ControlStructureDemo.java
1
2 public class ControlStructureDemo {
3
4 public static void main(String[] args) {
5
6
7 }
8
9 }
10
Problems
Javadoc
Declaration
Console
<terminated> Sample [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:05:30 pm)
Hello
Welcome
Writable
Smart Insert
5 : 9

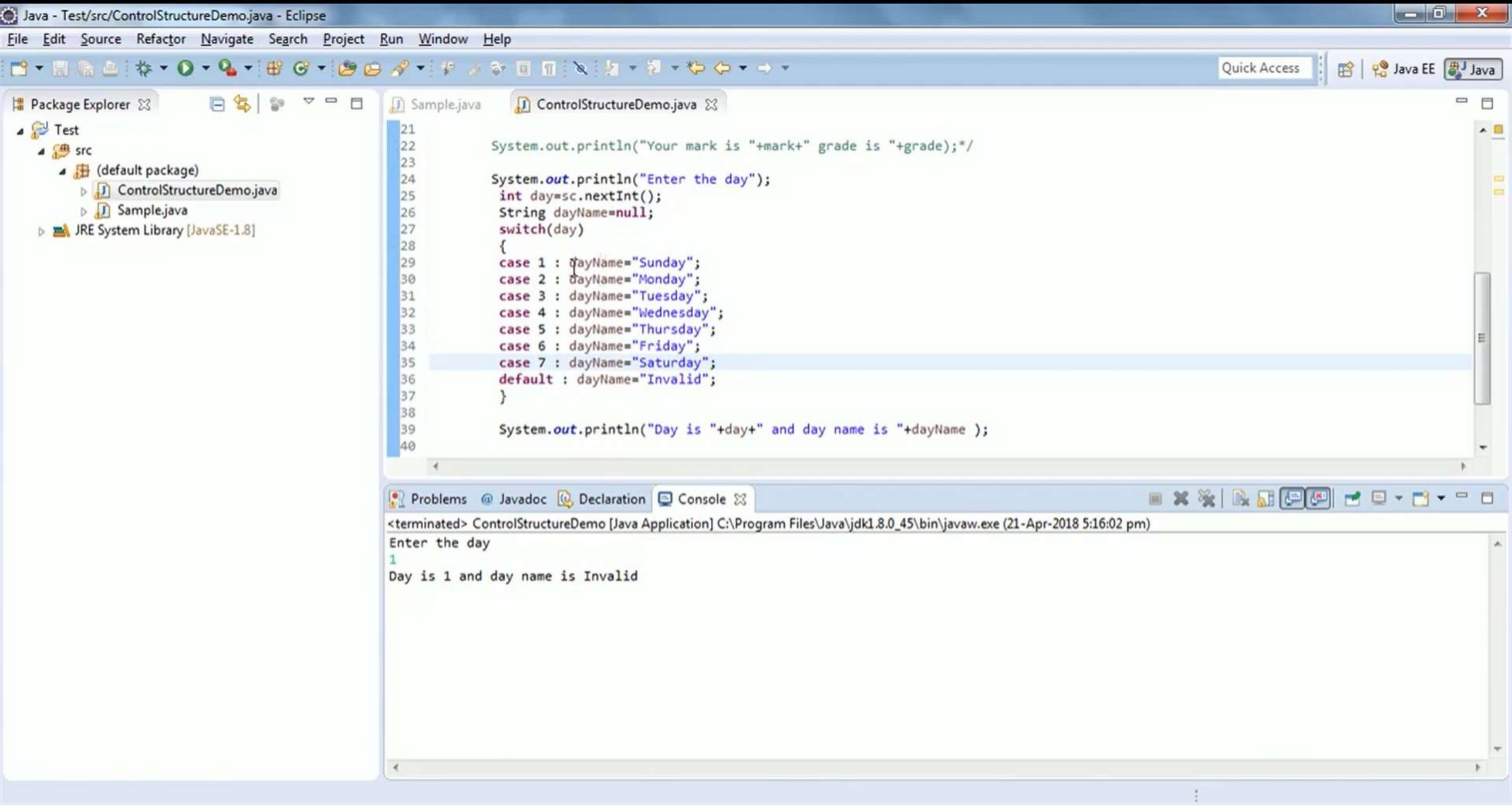

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
*Sample.java
ControlStructureDemo.java
5
6 public static void main(String[] args) {
7 int mark=0;
8
9 Scanner sc=new Scanner(System.in);
10 System.out.println("Enter the marks ");
11 mark = sc.nextInt();
12 char grade=' ';
13 if(mark>90)
14 grade='A';
15 else if(mark>80)
16 grade='B';
17 else if(mark > 70)
18 grade = 'C';
19 else
20 grade = 'D';
21
22 System.out.println("Your mark is "+mark+" grade is "+grade);
23
24
Problems
Javadoc
Declaration
Console
<terminated> Sample [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:05:30 pm)
Hello
Welcome
Writable
Smart Insert
10 : 45

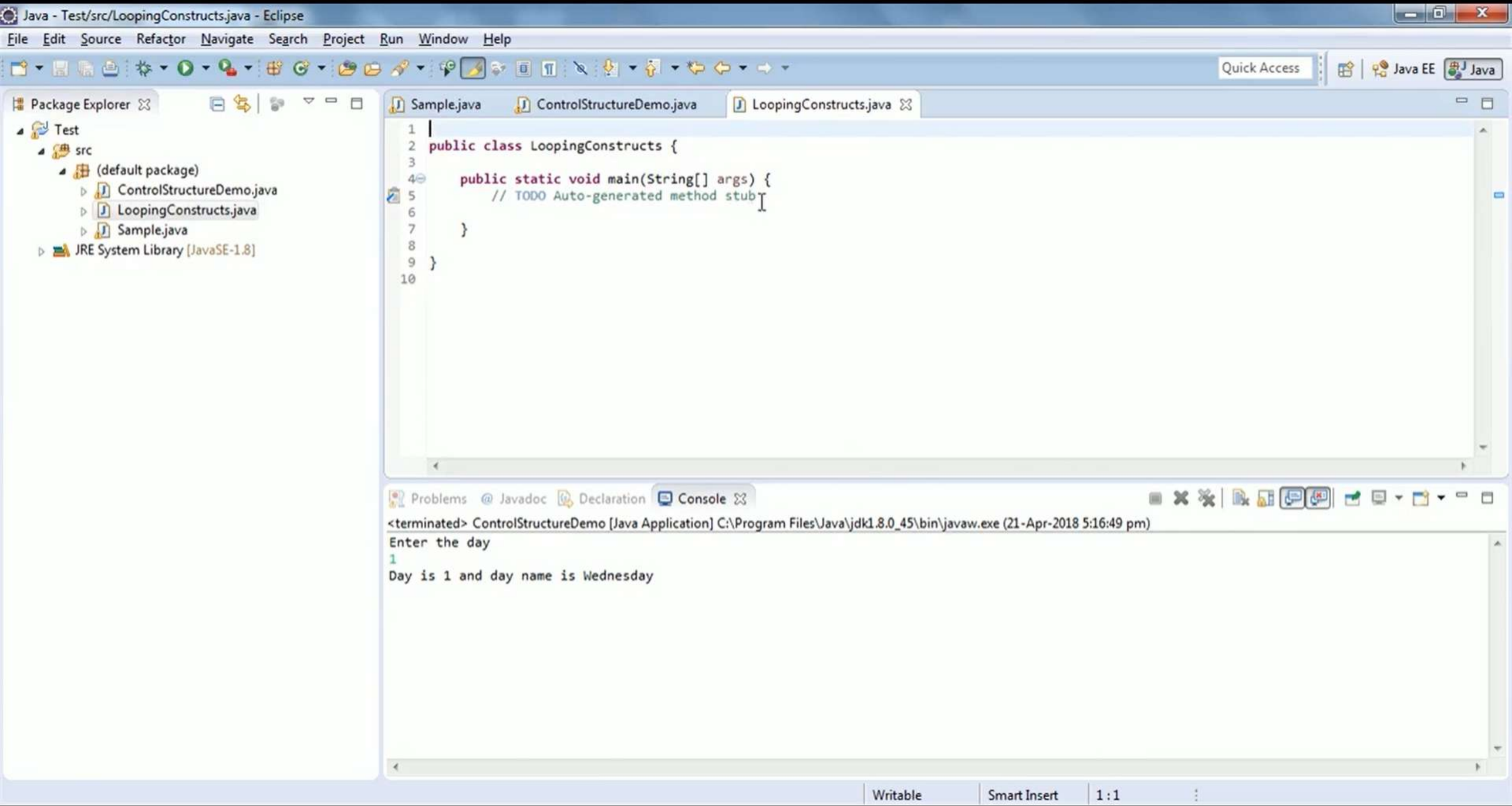

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
*Sample.java
ControlStructureDemo.java
5
6 public static vo
7 int mark=0;
8
9 Scanner sc=n
10 System.out.p
11 mark = sc.ne
12 char grade='
13 if(mark>90)
14 grade='A
15 else if(mark
16 grade='B
17 else if(mark
18 grade =
19 else
20 grade =
21
22 System.out.p
23
24
de);
Save and Launch
Select resources to save:
Sample.java
Select All
Deselect All
Always save resources before launching
OK
Cancel
Problems
Javadoc
No consoles to display at this time
Writable
Smart Insert
10 : 45
Launching ControlStructureDemo: (2%)

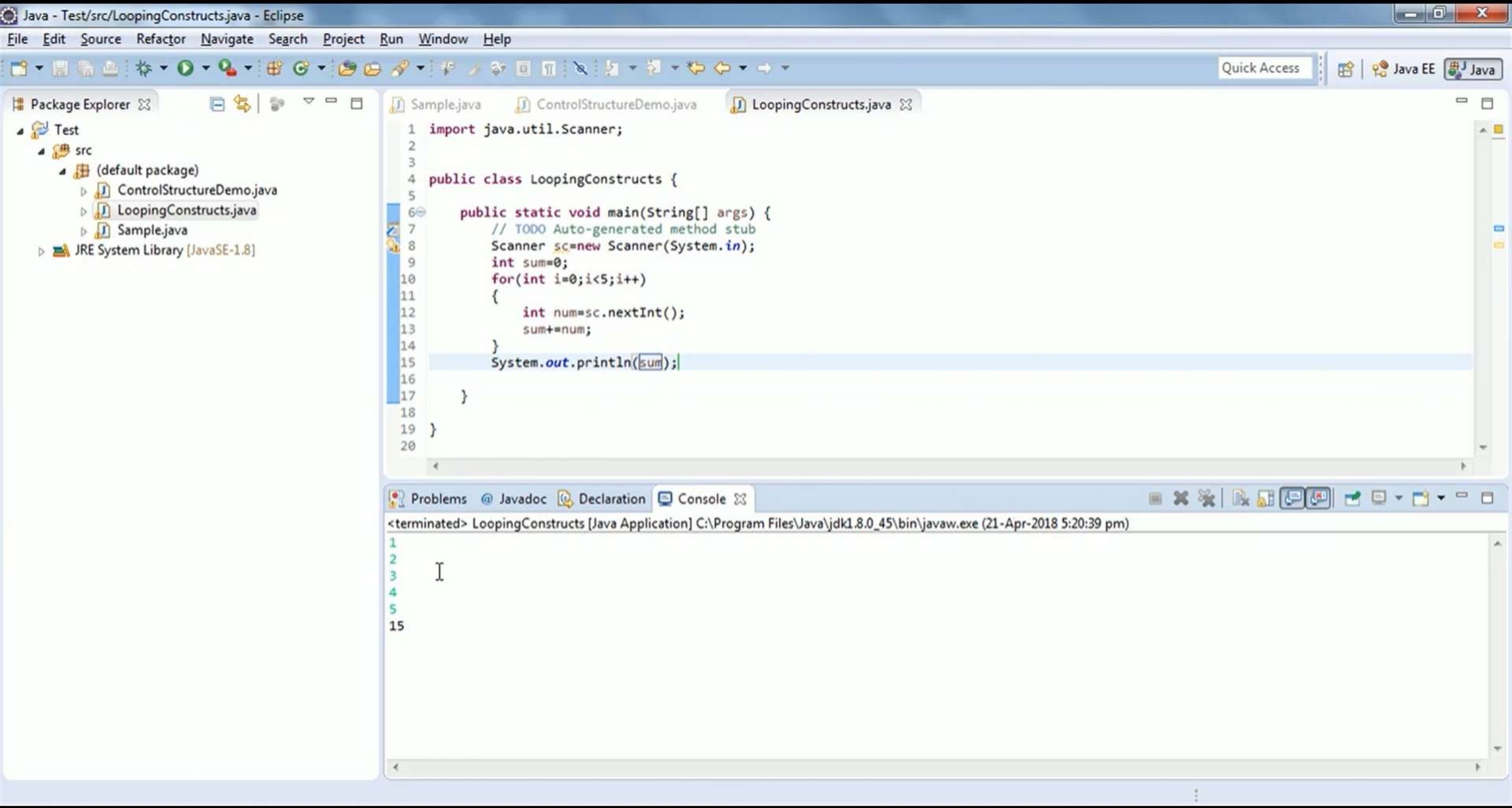

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
5
6 public static void main(String[] args) {
7 int mark=0;
8
9 Scanner sc=new Scanner(System.in);
10 System.out.println("Enter the marks ");
11 mark = sc.nextInt();
12 char grade=' ';
13 if(mark>90)
14 grade='A';
15 else if(mark>80)
16 grade='B';
17 else if(mark > 70)
18 grade = 'C';
19 else
20 grade = 'D';
21
22 System.out.println("Your mark is "+mark+" grade is "+grade);
23
24
Problems
Javadoc
Declaration
Console
<terminated> ControlStructureDemo [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:10:52 pm)
Enter the marks
95
Your mark is 95 grade is A

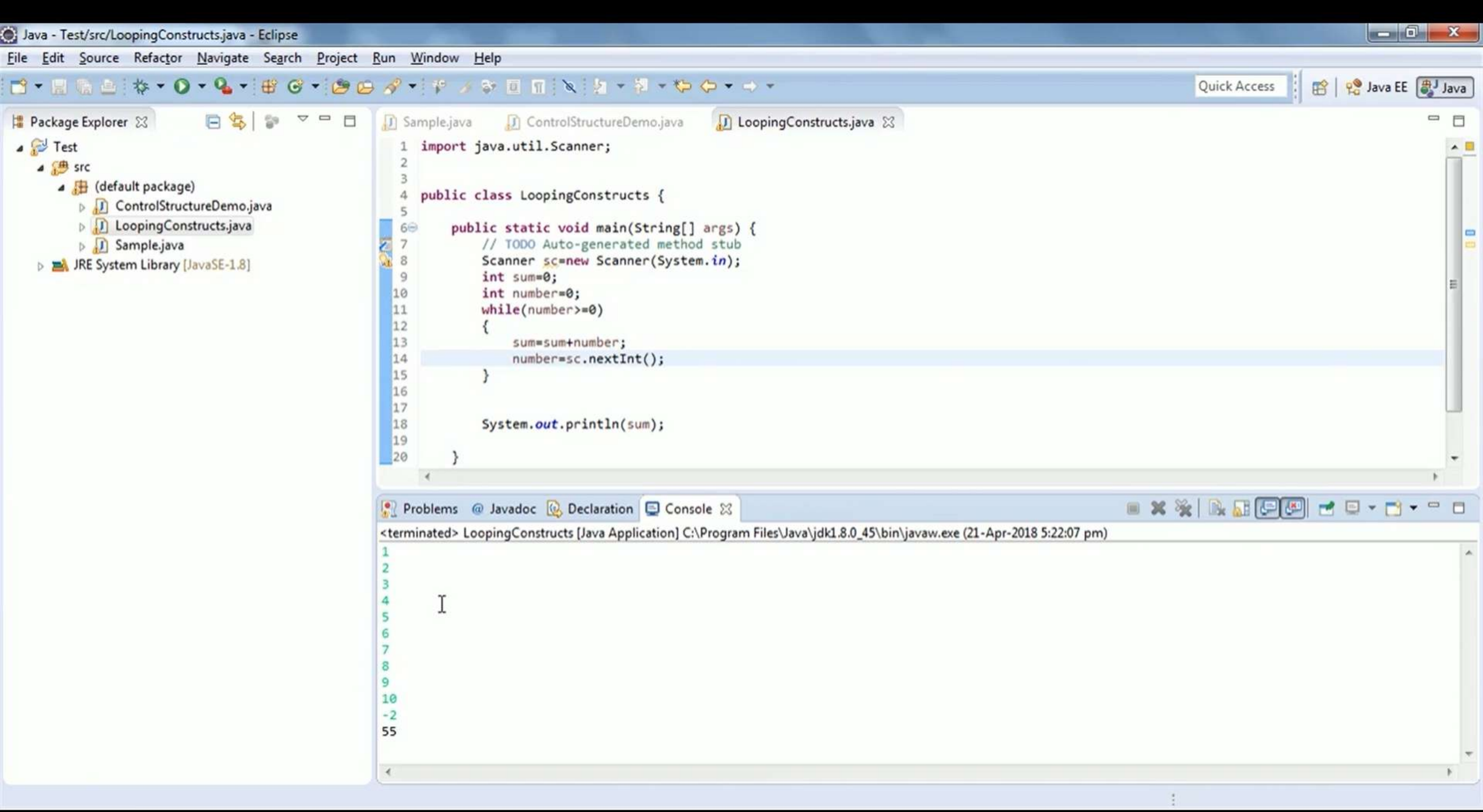

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
21
22 System.out.println("Your mark is "+mark+" grade is "+grade);*/
23
24 System.out.println("Enter the day");
25 int day=sc.nextInt();
26 String dayName=null;
27 switch(day)
28 {
29 case 1 : dayName="Sunday";break;
30 case 2 : dayName="Monday";break;
31 case 3 : dayName="Tuesday";break;
32 case 4 : dayName="Wednesday"; break;
33 case 5 : dayName="Thursday";break;
34 case 6 : dayName="Friday";break;
35 case 7 : dayName="Saturday";break;
36 default : dayName="Invalid";
37 }
38
39 System.out.println("Day is "+day+" and day name is "+dayName);
40
Problems
Javadoc
Declaration
Console
ControlStructureDemo [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:15:21 pm)
Writable
Smart Insert
25 : 15

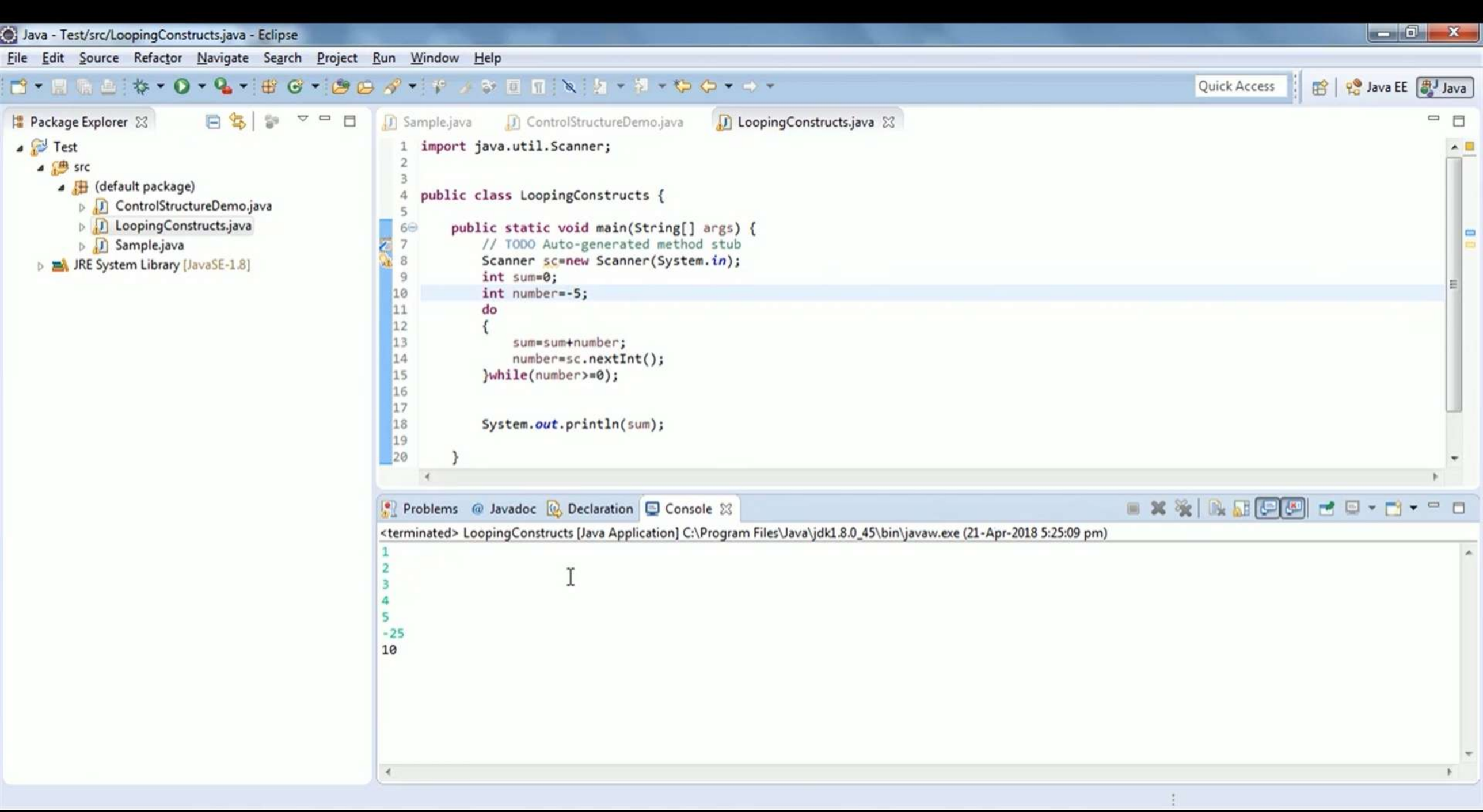

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
21
22 System.out.println("Your mark is "+mark+" grade is "+grade);*/
23
24 System.out.println("Enter the day");
25 int day=sc.nextInt();
26 String dayName=null;
27 switch(day)
28 {
29 case 1 : dayName="Sunday";break;
30 case 2 : dayName="Monday";break;
31 case 3 : dayName="Tuesday";break;
32 case 4 : dayName="Wednesday"; break;
33 case 5 : dayName="Thursday";break;
34 case 6 : dayName="Friday";break;
35 case 7 : dayName="Saturday";break;
36 default : dayName="Invalid";
37 }
38
39 System.out.println("Day is "+day+" and day name is "+dayName);
40
Problems
Javadoc
Declaration
Console
<terminated> ControlStructureDemo [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:15:21 pm)
Enter the day
6
Day is 6 and day name is Friday

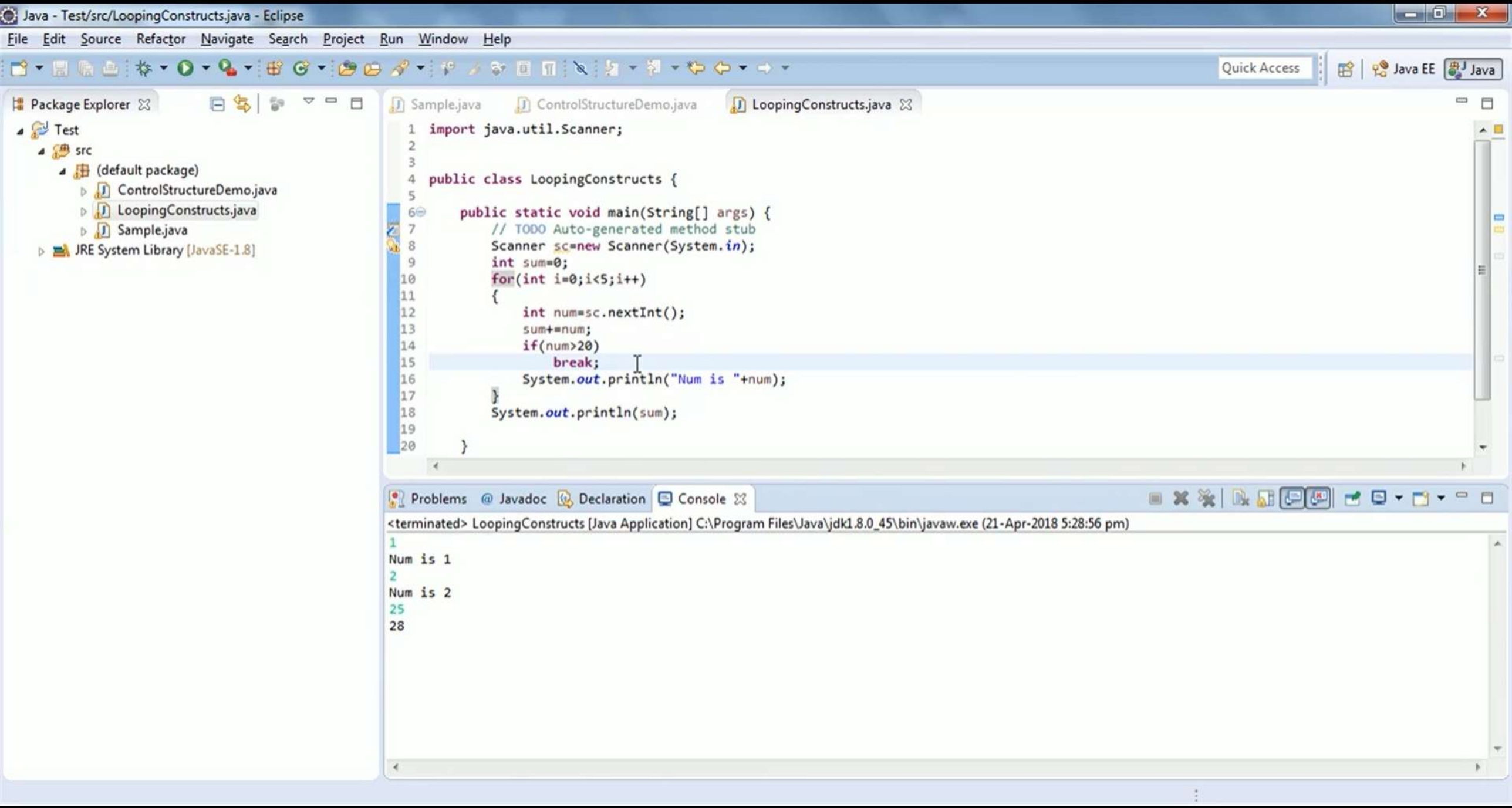

Java - Test/src/ControlStructureDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
21
22 System.out.println("Your mark is "+mark+" grade is "+grade);*/
23
24 System.out.println("Enter the day");
25 int day=sc.nextInt();
26 String dayName=null;
27 switch(day)
28 {
29 case 1 : dayName="Sunday";
30 case 2 : dayName="Monday";
31 case 3 : dayName="Tuesday";
32 case 4 : dayName="Wednesday";
33 case 5 : dayName="Thursday";
34 case 6 : dayName="Friday";
35 case 7 : dayName="Saturday";
36 default : dayName="Invalid";
37 }
38
39 System.out.println("Day is "+day+" and day name is "+dayName);
40
Problems
Javadoc
Declaration
Console
<terminated> ControlStructureDemo [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:16:02 pm)
Enter the day
1
Day is 1 and day name is Invalid


Java - Test/src/LoopingConstructs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
LoopingConstructs.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
LoopingConstructs.java
1
2 public class LoopingConstructs {
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6
7 }
8
9 }
10
Problems
Javadoc
Declaration
Console
<terminated> ControlStructureDemo [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:16:49 pm)
Enter the day
1
Day is 1 and day name is Wednesday
Writable
Smart Insert
1 : 1


Java - Test/src/LoopingConstructs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
LoopingConstructs.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
LoopingConstructs.java
1 import java.util.Scanner;
2
3
4 public class LoopingConstructs {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 Scanner sc=new Scanner(System.in);
9 int sum=0;
10 for(int i=0;i<5;i++)
11 {
12 int num=sc.nextInt();
13 sum+=num;
14 }
15 System.out.println(sum);
16
17 }
18
19 }
20
Problems
Javadoc
Declaration
Console
<terminated> LoopingConstructs [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:20:39 pm)
1
2
3
4
5
15


Java - Test/src/LoopingConstructs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
LoopingConstructs.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
LoopingConstructs.java
1 import java.util.Scanner;
2
3
4 public class LoopingConstructs {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 Scanner sc=new Scanner(System.in);
9 int sum=0;
10 int number=0;
11 while(number>=0)
12 {
13 sum=sum+number;
14 number=sc.nextInt();
15 }
16
17
18 System.out.println(sum);
19
20 }
Problems
Javadoc
Declaration
Console
<terminated> LoopingConstructs [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:22:07 pm)
1
2
3
4
5
6
7
8
9
10
-2
55


Java - Test/src/LoopingConstructs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
LoopingConstructs.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
LoopingConstructs.java
1 import java.util.Scanner;
2
3
4 public class LoopingConstructs {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 Scanner sc=new Scanner(System.in);
9 int sum=0;
10 int number=-5;
11 do
12 {
13 sum=sum+number;
14 number=sc.nextInt();
15 }while(number>=0);
16
17
18 System.out.println(sum);
19
20 }
Problems
Javadoc
Declaration
Console
<terminated> LoopingConstructs [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:25:09 pm)
1
2
3
4
5
-25
10


Java - Test/src/LoopingConstructs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
Test
src
(default package)
ControlStructureDemo.java
LoopingConstructs.java
Sample.java
JRE System Library [JavaSE-1.8]
Sample.java
ControlStructureDemo.java
LoopingConstructs.java
1 import java.util.Scanner;
2
3
4 public class LoopingConstructs {
5
6 public static void main(String[] args) {
7 // TODO Auto-generated method stub
8 Scanner sc=new Scanner(System.in);
9 int sum=0;
10 for(int i=0;i<5;i++)
11 {
12 int num=sc.nextInt();
13 sum+=num;
14 if(num>20)
15 break;
16 System.out.println("Num is "+num);
17 }
18 System.out.println(sum);
19
20 }
Problems
Javadoc
Declaration
Console
<terminated> LoopingConstructs [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:28:56 pm)
1
Num is 1
2
Num is 2
25
28

Java - Test/src/LoopingConstructs.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- Test
  - src
    - (default package)
      - ControlStructureDemo.java
      - LoopingConstructs.java
      - Sample.java
  - JRE System Library [JavaSE-1.8]

Sample.java | ControlStructureDemo.java | LoopingConstructs.java

```
1  import java.util.Scanner;
2
3
4  public class LoopingConstructs {
5
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8          Scanner sc=new Scanner(System.in);
9          int sum=0;
10         for(int i=0;i<5;i++)
11         {
12             int num=sc.nextInt();
13             sum+=num;
14             if(num>20)
15                 continue;
16             System.out.println("Num is "+num);
17         }
18         System.out.println(sum);
19
20     }
```

Problems | Javadoc | Declaration | Console

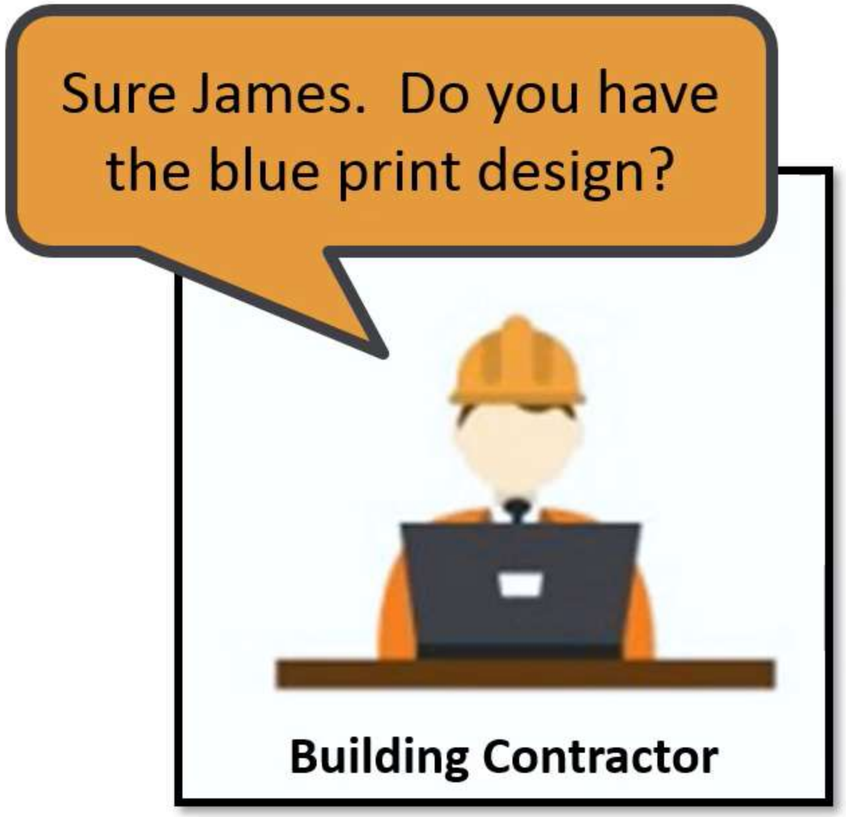
<terminated> LoopingConstructs [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (21-Apr-2018 5:29:34 pm)

```
1
Num is 1
2
Num is 2
23
3
Num is 3
4
Num is 4
33
```

TEKNOTURF

CLASSES AND OBJECTS

Overview


Sure James. Do you have the blue print design?
Building Contractor


I am planning to construct a house. Can you do it for me?

Overview

Great James! I will get it ready in a day or two.

Design Engineer

Hi Jacob. Can you give me a good design for the new house I am planning to construct?

Building Contractor

Overview


Building Contractor


Home Sweet Home


Multi-Corporations inc.

In this module you will learn -

- Classes and Objects
- Access Specifiers
- Usage of Getters and Setters

Classes and Objects

TEKNOTURF

What is an Object?

- Any real world entity with well defined properties is called an object.
- Object will have a state, behaviour and a unique identity
- Objects can be tangible (physical entity) or intangible (cannot be touched or felt)

A wooden chair

A chemical process

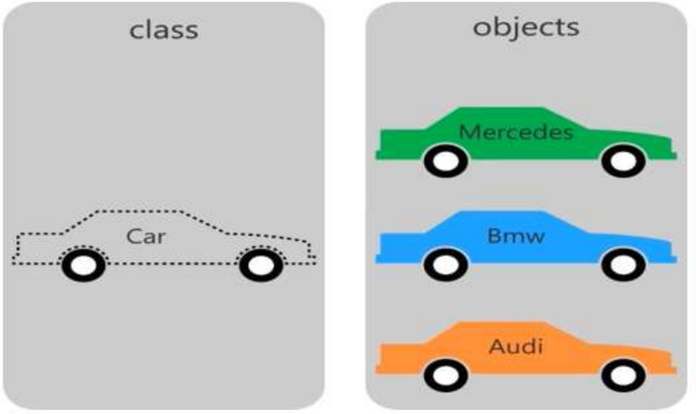
A transaction

Classes and Objects

TEKNOTURF

What is a class?

- A *class* is a template or *blue print* that describes the behaviors/states that an object of its type supports
- A class defines its
  - object's properties and
  - object's behavior through methods



Classes and Objects

TEKNOTURF

Class is a template for a collection of objects that share a common set of attributes and behaviour.

In the perspective of OO programming

- Class is a description of the object
  - It describes the data that each object possesses
  - It also describes the various behaviors of the object

Object is an instance of a class

Classes and Objects

Class: Employee

Employee ID : 101
Name : Rohith
Salary : 25000.00

Employee: Object

```
class Employee
{
   //Attributes
    int empID;
    String name;
    double salary;

    //Methods
    void punchCard()
    {
        //do something
    }
    void  doProject()
    {
       //do something
    }
}

}
```

Classes and Objects

Car Object

Attributes :
model = “Mustang”
Color = “Yellow”
numOfPassengers = 0
amtOfGas = 16.5

Behaviour:
Add / Remove Passenger
Get Tank Filled
Report when out of Gas

```
class Car
{
  //Attributes
  String model;
   String color;
   int numOfPassengers;
   float amtOfGas;

   //Methods
   void addPassenger()
   {
     // do something
    }
```

```
void removePassenger()
   {
     // do something
    }
 void fillTank()
   {
     // do something
    }
 void report_OutOfGas()
   {
     // do something
    }
}
```

Class Declaration

TEKNOTURF

Declaring a class

```
<modifier>* class <class_name> {
        <attribute_declaration>*
        <constructor_declaration>*
        <method_declaration>*
}
```

Example:

```
public class Employee {
    private int empId;
    private String name;
    private  double salary;
    public void setEmpId(int  id) {
       empId = id;
    }
}
```

Attribute Declaration

TEKNOTURF

Attributes/Fields are used to declare the properties of a class. These attributes are called instance variables.

They can be intrinsic types (int, boolean…) or user-defined type.

Basic syntax of an attribute :

- <visibility>* <modifier>* <type> <name> [= <initial_value>]

Example: private double salary;

Method Declaration

Methods describe the responsibility of the class. These methods are common for all the objects. Hence they are known as **instance methods.**

Method will have a name and a return type.

Methods can accept parameters.

Basic syntax of a method:

```
<visibility>* <modifier>* <return_type> <name> ( <argument>* ) {
                    <statement>*
        }
```

Example :

```
public void calculateArea(int side) {
        int area = side * side;
    System.out.println(area);
    }
```

```
public boolean isAgeValid( ) {
    int age = 15;
    if(age>0 && age <=100)
            return true;
    else
            return false;
}
```

Create Object

TEKNOTURF

Creation of Object

- Objects are created for a class using the keyword new.
- In Java, all Objects are created at run time in the heap area.

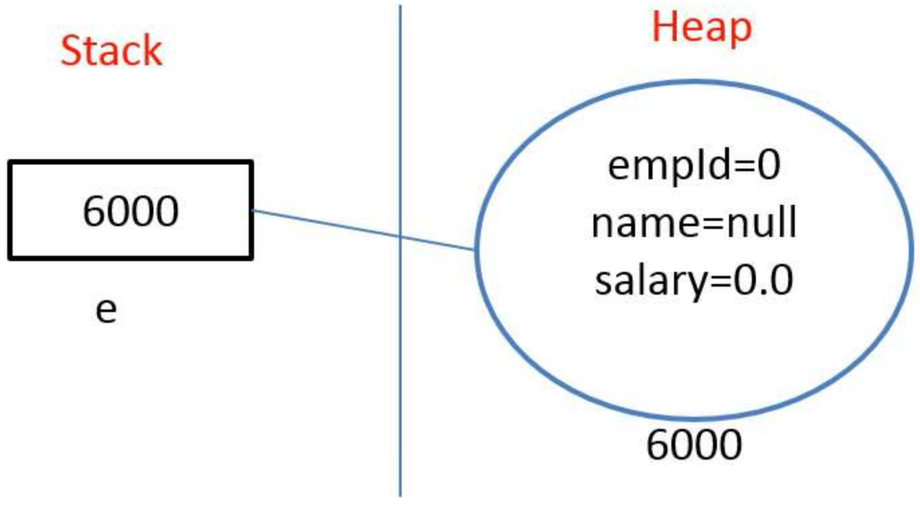
Syntax

- class_name reference_variable = new class_name(<parameter list>)

Example

- Employee e=new Employee();

Here variable ‘e’ is the name of the object. ‘e’ is the reference variable that holds the address of the Object created in Heap.



Note : Instance variables will be assigned with default values.

Create Object and Access Members

TEKNOTURF

- The attributes and methods of a class can be accessed using the dot operator as <object>.<member>

```
//object creation
    Employee empObj=new Employee();

    //assign meaningful state
    empObj.empId(101);
    empObj.name("Rohit");
    empObj.salary(25000);

    //retrieve values
    System.out.println("Emp ID  "+empObj.empId);
    System.out.println("Name  "+empObj.name);
    System.out.println("Salary  "+empObj.salary);
```

class Example

```
public class Employee {

  //Attributes
        int empId;
        String name;
        double salary;

  //Methods

   public void calculateGrossPay(int bonus)
  {
      return salary + bonus
  }
}
```

```
public class EmployeeMain
{
   public static void main(String a[])
   {
      //object creation
         Employee empObj=new Employee();

      //assign meaningful state
          empObj.empId=101;
          empObj.name="Rohit";
          empObj.salary=25000;
     //retrieve values
          System.out.println("Emp ID  "+empObj.empId);
          System.out.println("Name  "+empObj.name);
          System.out.println("Salary  "+empObj.salary);
     }
}
```

Accessors and Mutators

TEKNOTURF

Encapsulation is an important OO Concept.

How is encapsulation implemented in Java?

- By wrapping up of data and methods, that operate on that data into a single unit, class
- Data inside a class needs to be secure. Declare the data in a class as private.
- No outside class can access private data members of other class.
- Outside classes should be able to read and update the private data members only by using the public methods.
- These methods are called Accessors or Getters and Mutators or Setters.
- Data Hiding is not possible without Encapsulation.

Accessors and Mutators

TEKNOTURF

Accessor

- Method used to return the value of a private field
- Does not change the state of the Object
- Example :

```
public int getSalary() {
    return salary;
}
```

Mutator

- Method used to set a value of a private field
- Changes the state of the Object
- Example :

```
public void setSalary(double  sal) {
    salary = sal;
}
```

Create Object and Access Members

TEKNOTURF

```
//object creation
    Employee empObj=new Employee();

    //assign meaningful state using Setters
    empObj.setEmpId(101);
    empObj.setName("Rohit");
    empObj.setSalary(25000);

    //retrieve values using Getters
    System.out.println("Emp ID  "+empObj.getEmpId());
    System.out.println("Name  "+empObj.getName());
    System.out.println("Salary  "+empObj.getSalary());
```

class Example

TEKNOTURF

```
public class Employee {

//Attributes
private int empId;
private String name;
private double salary;

//Accessors  or  Getters
public int getEmpId() {
return empId;
}

public String getName() {
return name;
}

public double getSalary() {
return salary;
}

//Mutators or Setters
public void setEmpId(int id) {
    empId = id;
}
public void setName(String name1) {
    name = name1;
}
public void setSalary(double sal) {
     salary = sal;
}
}
```

Create Object and Access Members

TEKNOTURF

```
public class EmployeeMain
{
    public static void main(String a[])
    {
    //Create Object
        Employee empObj=new Employee();

        //assign meaningful state
        empObj.setEmpId(101);
        empObj.setName("Rohit");
        empObj.setSalary(25000);

        //retrieve values and print
        System.out.println("Emp ID  "+empObj.getEmpId());
        System.out.println("Name  "+empObj.getName());
        System.out.println("Salary  "+empObj.getSalary());
    }
}
```

EmployeeMain class depicts how to create an object for the Employee class. It also shows how to access the members of the class using that object.

Access Specifier

Information Hiding is implemented using Access Specifiers known as visibility modifiers.

They regulate access to classes, fields and methods.

They specify if a field or method in a class can be invoked from another class or sub-class.

In simple words, they can be used to restrict access.

Types Of Access Specifiers in Java

- public
- private
- protected
- default(no specifier)

Access Specifier

public

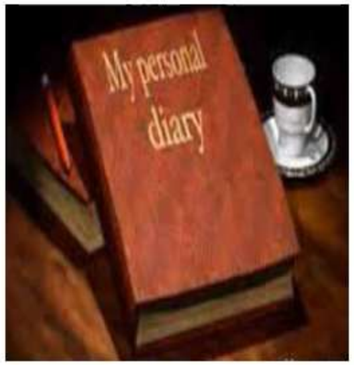
- Classes, methods, and fields declared as public can be accessed from any class in an application

default

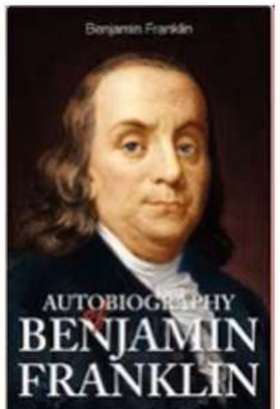
- If classes, variables, and methods have a default specifier we can access them only within the same package, but not from outside this package

private

- Private methods and fields can only be accessed within the same class to which the methods and fields belong



private



public

default

- When no access specifier is specified for an element, its accessibility level is default.
- There is no keyword default.

Access Specifier

Access Modifiers	default	private	protected	public
Accessible inside the class	yes	yes	yes	yes
Accessible within the subclass inside the same package	yes	no	yes	yes
Accessible outside the package	no	no	no	yes
Accessible within the subclass outside the package	no	no	yes	yes

Let us discuss about protected, subclass and packages later

Summary

- Classes and Objects
- Access Specifiers
- Usage of Getters and Setters

TEKNOTURF

CONSRUCTORS

In this module you will learn -

TEKNOTURF

- Constructor – Default and Parameterized
- ‘this’ reference
- Constructor Overloading
- Method Overloading

Constructor - Overview

TEKNOTURF

How can we create an object of the employee shown here using Java?

Employee e=new Employee()

But the id is 0, name is null and salary is 0.0.
It is not valid.
How to create the object with proper state?

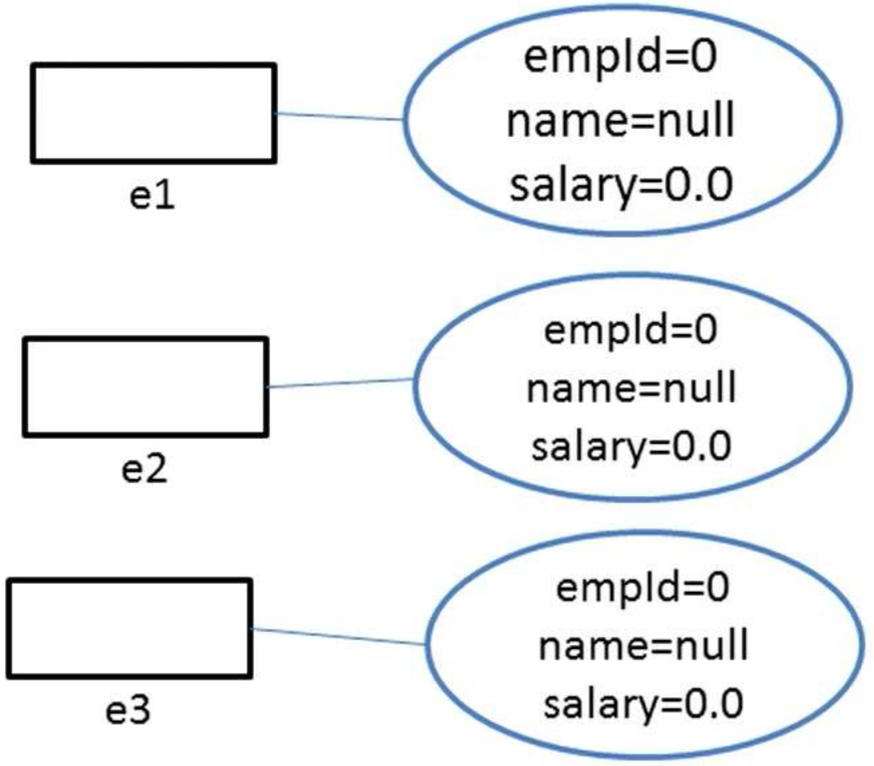
Employee ID : 101
Name : Rohith
Salary : 25000.00

Let us discuss this concept now.

Constructor

- Consider the following statements

```
Employee e1=new Employee();
Employee e2=new Employee();
Employee e3=new Employee();
```



Note :

- Objects are created.
- They are not in proper state.
- Instance variables are allotted the default value.

Constructor

TEKNOTURF

Constructor is a special method invoked implicitly when an object is created

It initializes the instance variables with proper value

Rules for writing a constructor

- Constructor name must be the same as the name of the class
- Constructors must not have a return type (not even void)

Types of Constructors

- Default Constructor
- Parameterized Constructor

Default Constructor

TEKNOTURF

Default Constructor (no argument constructor)

A constructor with no arguments is known as a default constructor

Example

```
public class Employee   {
   private int empId;
  private String name;
  private float salary;

  public Employee()   {
       System.out.println("Default Constructor");
  }
  public static void main(String a[])
 {
        Employee e=new Employee();
  }
}
```

Default Constructor

Output : Default Constructor

Parameterized Constructor

TEKNOTURF

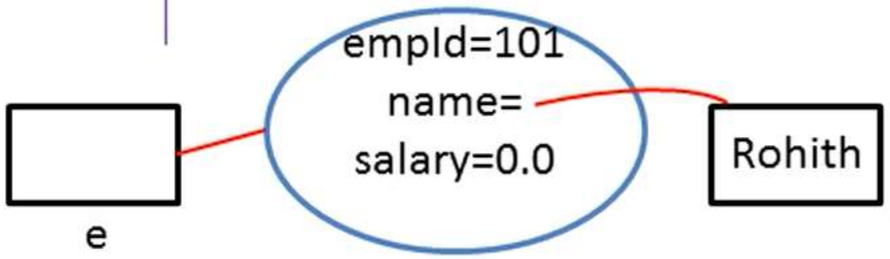
Parametrized Constructor

A constructor with an argument list is known as a parameterized constructor.

Example

```
public class Employee   {
 private int empId;
 private String name;
 private float salary;

//Constructor
public Employee(int id,String ename)   {
      System.out.println("In Parametrized Constructor");
     empId = id;
      name = ename;
 }
  public static void main(String a[])
  {
        Employee e=new Employee(101,"Rohith");
   }
}
```

Output : In Parametrized Constructor



Constructors

TEKNOTURF

- If a constructor is not written explicitly for a class, what happens when an object is created?
  - Java compiler builds a default constructor for that class and initiates the fields with the default value.
- When a constructor is explicitly written by the developer, the compiler will not provide the default constructor.
- If a constructor with parameters is declared and we want to create an object using no argument constructor, we must explicitly write a no argument constructor.

“this" keyword

```
class  Employee{
   int empId;
   String name;

   public Employee(int empId,String name)
   {
      empId = empId;
      name = name;
   }
   void display()
   {
       System.out.println(“ID  : “+empId+“ Name : "+name);
    }

    public static void main(String args[])
    {
        Employee  e1= new Employee(101,“Tom");
        e1.display();
    }
}
```

Output of this code :

ID : 0 Name : null

The reason for this output is: parameter and instance variables have the same name

The solution for this problem is to use the keyword “this”

“this" keyword

“this” is a reference variable that refers to the current object

Example : Usage of “this” in constructor for initializing the attributes

```
class  Employee{
   int empId;
   String name;

   public Employee(int empId,String name)
   {
      this.empId = empId;
      this.name = name;
   }
   void display()
   {
       System.out.println(“ID  : “+empId+“ Name : "+name);
    }
```

```
    public static void main(String args[])
    {
       Employee  e1= new Employee(101,“Tom");
       e1.display();
    }
}
```

Output of this code :

ID : 101 Name : Tom

“this” keyword

TEKNOTURF

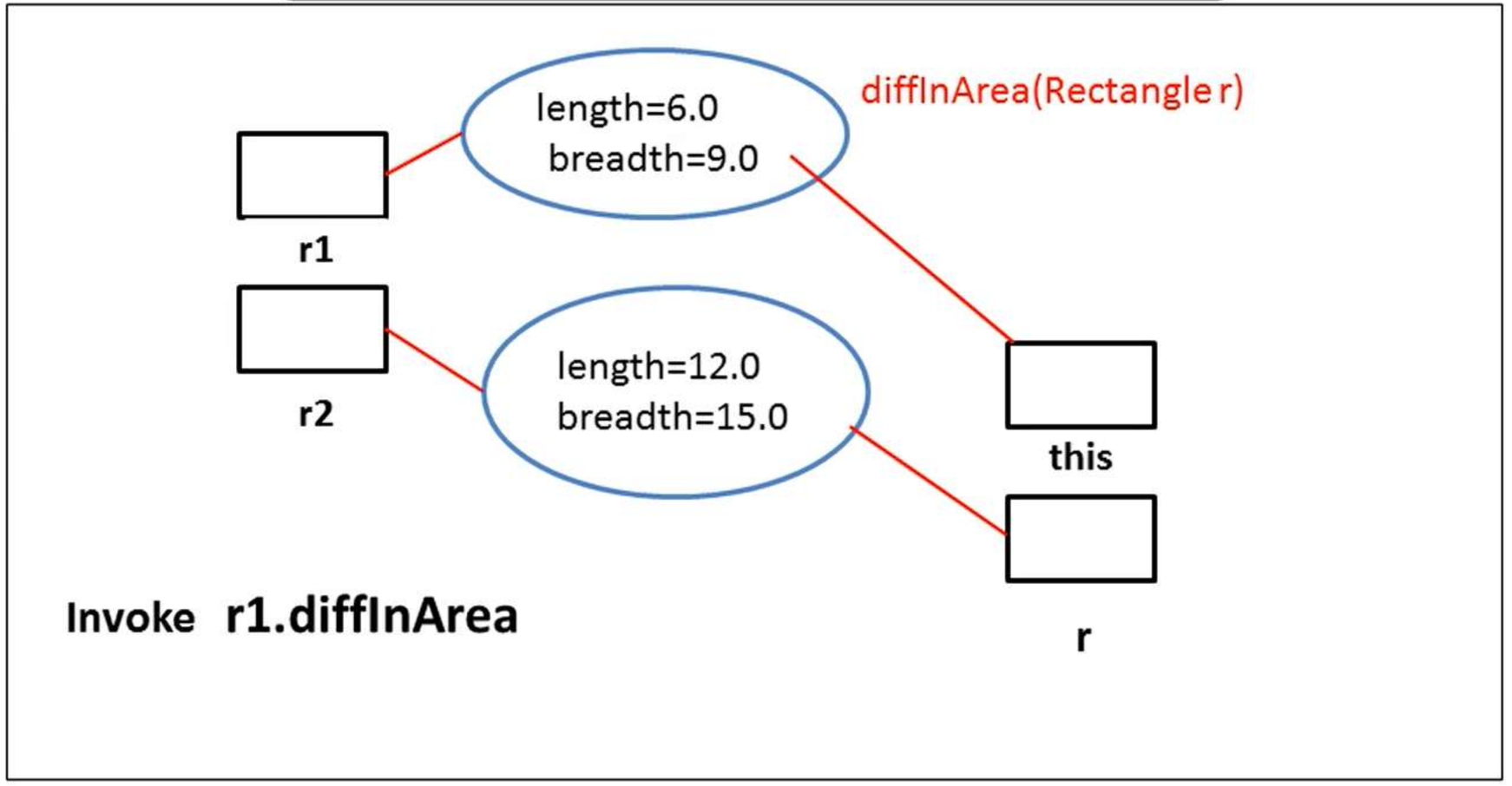
Example : Usage of “this” in method

```
public class Rectangle {
   private float length;
   private float breadth;

   //Constructor
   public Rectangle(float length, float breadth)
   {
        this.length = length;
        this.breadth = breadth;
   }
   //Function to calculate Area
   public float calcArea()
   {
        return length*breadth;
   }
```

```
//difference between the area of 2 rectangle
public float diffInArea(Rectangle r)
{
        float area_difference =
                Math.abs(this.calcArea() -
                                r.calcArea());
        return area_difference;
}
public static void main(String a[])
{
   Rectangle r1=new Rectangle(6,9);
   Rectangle r2=new Rectangle(12,15);
   float diffInArea=r1.diffInArea(r2);
   System.out.println("Difference in Area : “
                                +diffInArea);
}
}
```

“this” keyword

Example : Usage of “this” in method



Constructor Overloading

TEKNOTURF

A class can have any number of constructors but they should differ in the parameter list.

This technique is called constructor overloading.

The parameter list should vary in any of the following:

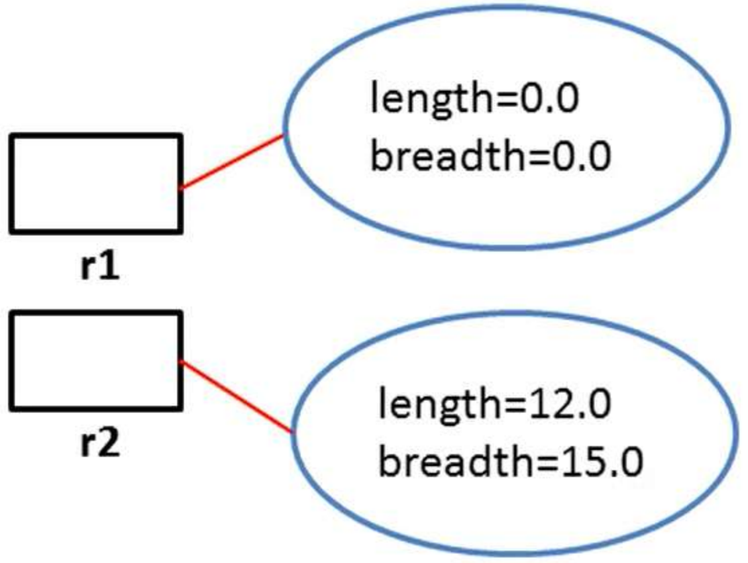
- Number of parameters
- Type of parameter
- Order of parameter

The compiler identifies which constructor is to be invoked based on the number of parameters in the list and their type.

Constructor Overloading

```
public class Rectangle {
private float length;
private float breadth;

//Constructor Overloading

public Rectangle()
{
    length=0;
    breadth=0;
}
public Rectangle(float length, float breadth)
{
  this.length = length;
  this.breadth = breadth;
}
```

```
public static void main(String a[])
{
    Rectangle r1=new Rectangle();
    Rectangle r2=new Rectangle(12,15);
}
}
```



Invoking a Constructor

TEKNOTURF

- Constructor is invoked implicitly when an object is created.
- It cannot be invoked explicitly.
- Within a class, one constructor can call another constructor using "this" keyword.
- Example:

```
public class Employee {

private int empId;
private String name;
private double salary;
private String panNo;

public Employee(int empId, String name, double salary)
{
    this.empId = empId;
    this.name = name;
     this.salary = salary;
}
```

```
public Employee(int empId, String name,
          double  salary, String panNo)
{
   this(empId,name,salary);
   this.panNo = panNo;
}
}
```

Note : this() should be the first statement in the constructor

Method Overloading

- In a class, if there is more than one method with the same name, but differing in parameters, those methods are called *overloaded methods*. This concept is called *Method Overloading*.


```
sing_a_Song()
{
    smooth and clear
}
```

```
sing_a_Song(illness)
{
   lots of disturbance
}
```

Method Overloading

TEKNOTURF

Rules for Method Overloading

- Methods should have same name but different arguments list
- Argument list should differ in
  - *Number of parameters*
  - *Type of parameter*
  - *Order of parameter*
- Return Type of the method is not to be considered

Method Overloading

TEKNOTURF

Example

```
public int add(int num1,int num2)
{
   return num1+num2;
}
public int add(int num1,int num2,int num3)
{
   return num1+num2+num3;
}
public String add(String s1,String s2)
{
   return s1+s2;
}
```

Number of Arguments are the same, but vary in the type

Vary in Number of Arguments

Method Overloading

TEKNOTURF

Valid overloaded methods

1. public void calculateArea(int a,int b) { }
   public void calculateArea(float a,float b) { }
   public void calculateArea(int a) { }
   public void calculateArea(int a,float b) { }

2. public void calculateArea(int a,int b) { }
   public void calculateArea(float a,float b) { }

3. public void calculateArea(int a,float b) { }
   public void calculateArea(float a,int b) { }

In-valid overloaded methods

1. public void calculateArea(int a) { }
   public int calculateArea(int a) { }

Not valid because the number of arguments is same and the change is only in the return type.

2. public void calculateArea(int a) { }
   public void calculateArea(int x) { }

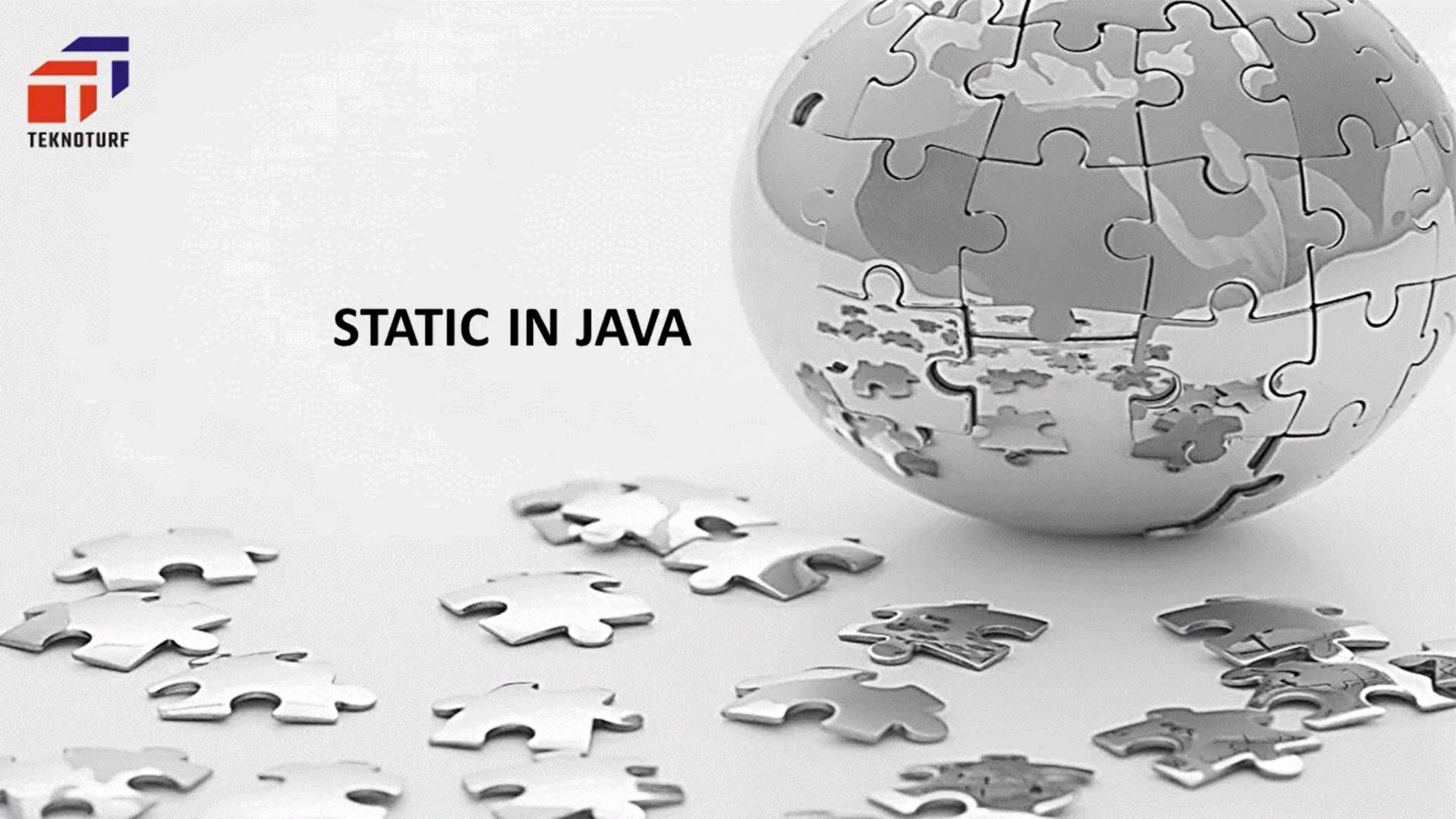
Not valid because the signature is same

Summary

- Constructor – Default and Parameterized
- ‘this’ reference
- Constructor Overloading
- Method Overloading

TEKNOTURF

STATIC IN JAVA



In this module you will learn -

- Usage of static keyword
- static variable and methods
- static block
- Initialization block

TEKNOTURF

Static

TEKNOTURF

This time, James plans to buy an apartment.

Static

This time, James plans to buy an apartment.

Static

TEKNOTURF

Certain things are available for each individual apartment, whereas certain things are common for all apartments.

Likewise, in a class, certain attributes will be applicable for each instance of that class and some attributes will be common to all instances in a class. These common attributes are termed ‘static’.

static

TEKNOTURF

When an object is created for a class, it will have its own copy of variables, called instance variables.

Certain fields are common to all objects in a class. We call these variables as class variables.

These variables are also called static variables because they are declared with the keyword static.

Static keyword is used as a modifier on variables, methods, block and nested classes.

A static modifier associates an attribute or method to the class as a whole, rather than as any particular instance of that class.

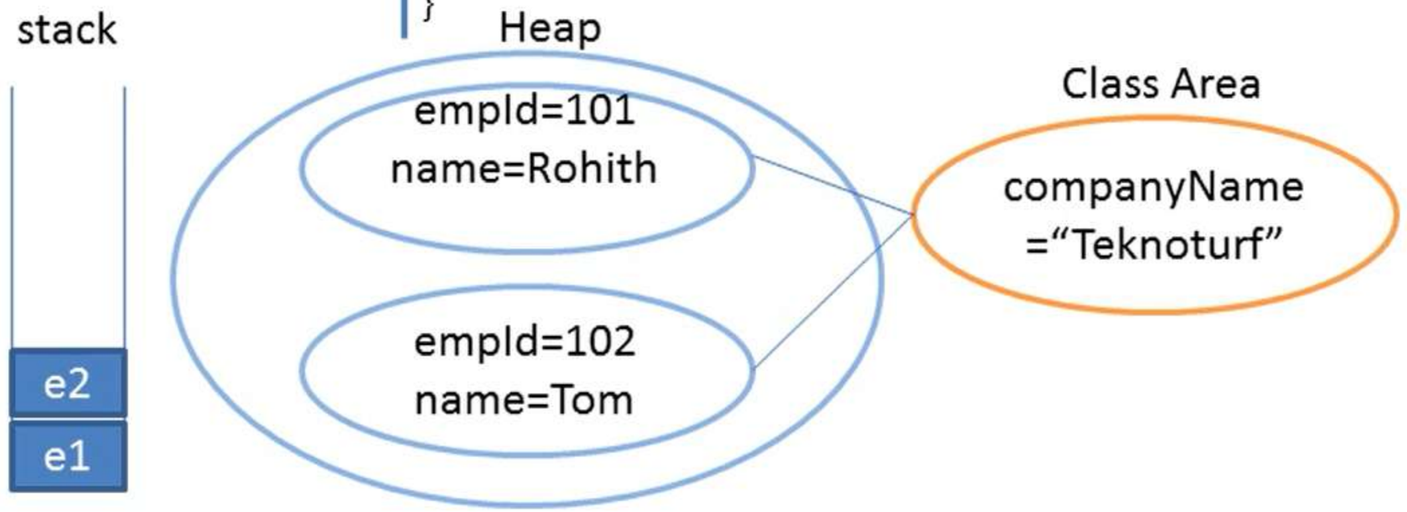
To access an instance member, an object is required. Whereas, static members can be accessed directly using the class name itself as className.staticmember

“this” cannot be referenced from a static context.

static variable

```
class Employee
{
  private int empId;
  private String name;
  private static String companyName="Teknoturf"

  //Constructor
  public Employee(int empId,String name)  {
       this.empId=empId;
       this.name=name;
}
```

```
public static void main(String a[])
  {
      Employee e1=new Employee(101,"Rohith");
      Employee e2=new Employee(102,"Tom")
      System.out.println("Company Name is "+
                          Employee.companyName);
      //This will also work
      //System.out.println("Company Name is "+
                          e1.companyName);
  }
}
```



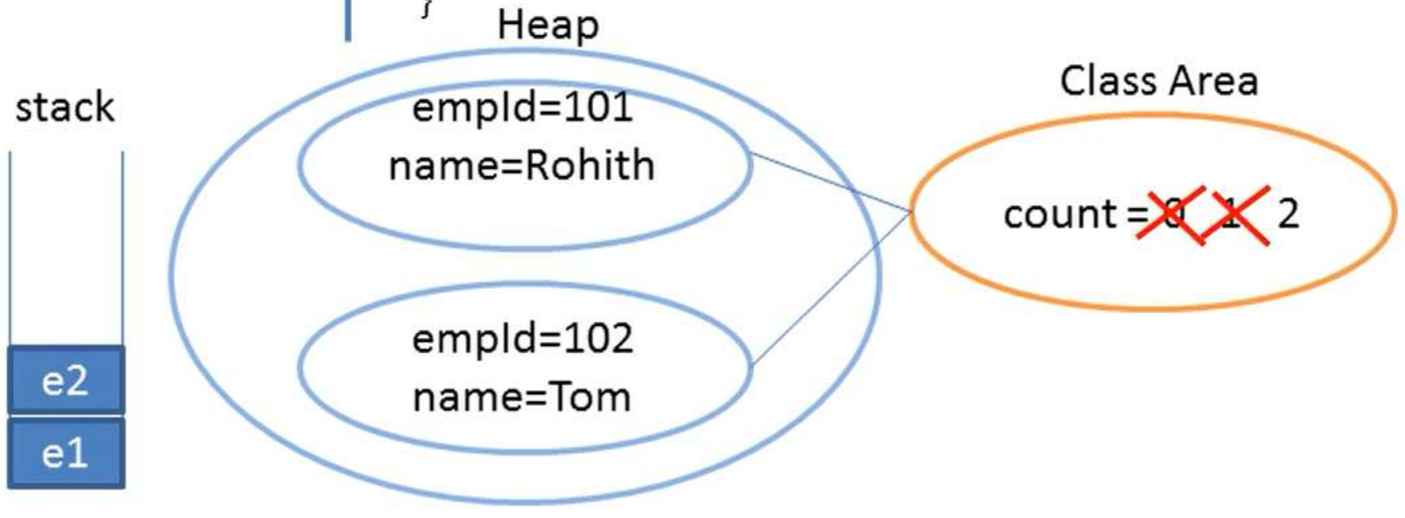
static variable

TEKNOTURF

Usage of static for count of objects created

```
class Employee
{
  private int empId;
  private String name;
  private static int count;

  //Constructor
  public Employee(int empId,String name)  {
       this.empId=empId;
       this.name=name;
       count++;
}
```

```
public static void main(String a[])
 {
      Employee e1=new Employee(101,"Rohith");
      Employee e2=new Employee(102,"Tom")
      System.out.println("Total Objects Created :  "+
                                    Employee.count);      //2
      //This will also work
      //System.out.println("Total Objects created : "+
              e1.count + " "+e2.count);   //2  2
  }
}
```



static method

```
public class Employee {

  private int empId;
  private String name;
  private static int count;

  //Constructor
  public Employee(int empId,String name)
  {
        this.empId=empId;
        this.name=name;
        count++;
  }
  public void setEmpId(int empId)
  {
         this.empId = empId;
  }
```

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

public int getEmpId()
{
   return empId;
}
public String getName()
{
   return name;
}
public static int displayCount()
{
   return count;
}
}
```

static method can access only static members

static method

```
public class EmployeeMain
{
    public static void main(String a[])
    {
        Employee e1=new Employee(101,"Rohith");
        Employee e2=new Employee(102,"Tom");
        System.out.println("Count of objects : "+Employee.displayCount()); //Count of objects : 2
     }
}
```

Static method accessed directly using class name

static

TEKNOTURF

A static method can access only other static members of a class.

A non-static method can access all members i.e., both static and non static members of a class.

The main() method in java is public static.

public access specifier denotes anyone can access/invoke it, such as JVM.

static keyword allows main() to be called before an object of the class has been created.

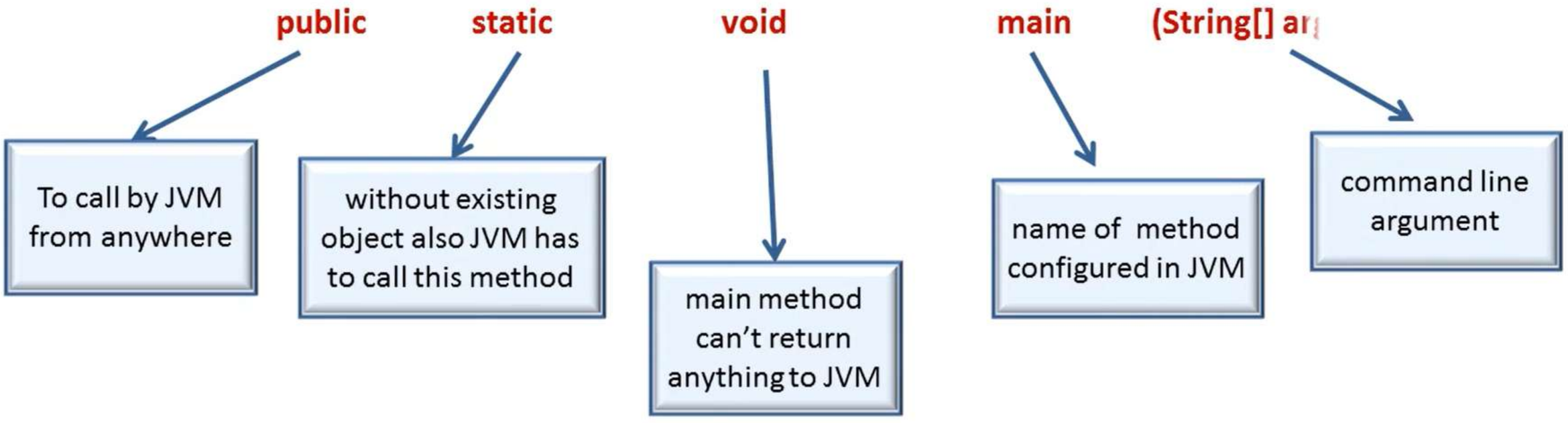
main() is called by the JVM before any objects are made. As it is static, it is directly invoked from class.

static

TEKNOTURF

The main method is declared as static

public static void main(String a[])



static block

TEKNOTURF

- Static variables can be initialized using the static block
- Static block gets executed when the class gets loaded in the memory
- There can exist multiple static blocks. They get executed in the same order in which they are written in the code

```
class StaticBlockDemo{
    static String language;
    static {
            language="Java";
    }
    public static void main(String a[])  {
            System.out.println("Language "+language);
    }
}
```

Initialization block

TEKNOTURF

- Instance Initialization Blocks are used to initialize instance variables
- Initialization blocks are executed whenever an instance is created for that class and before constructors are invoked.

```
public class Employee {
        {
                System.out.println("In Initialization block");
        }
        public Employee() {
                System.out.println("In constructor");
        }
        public static void main(String arg[])  {
                Employee e=new Employee();
        }
}
```

Output

```
In Initialization block
In constructor
```

Summary

- Usage of static keyword
- static variable and methods
- static block
- Initialization block

TEKNOTURF

WRAPPER CLASS

In this module you will learn -

- Wrapper class and its usage
- Convert primitive to wrapper and vice versa
- Convert String to primitive
- Autoboxing and unboxing concept

TEKNOTURF

Wrapper Class

TEKNOTURF

Java has eight primitive data types - byte, short, int, long, char, float, double, boolean

Object representation of primitive data types are called wrapper classes.

The java.lang package contains wrapper classes that corresponds to each primitive type

Certain concepts in java, like Collection work on objects.
Primitives can be used in those concepts using wrapper classes.

Wrapper Class

Wrapper class wraps around a data type and gives it an object appearance

Use this object wherever primitive is required as object

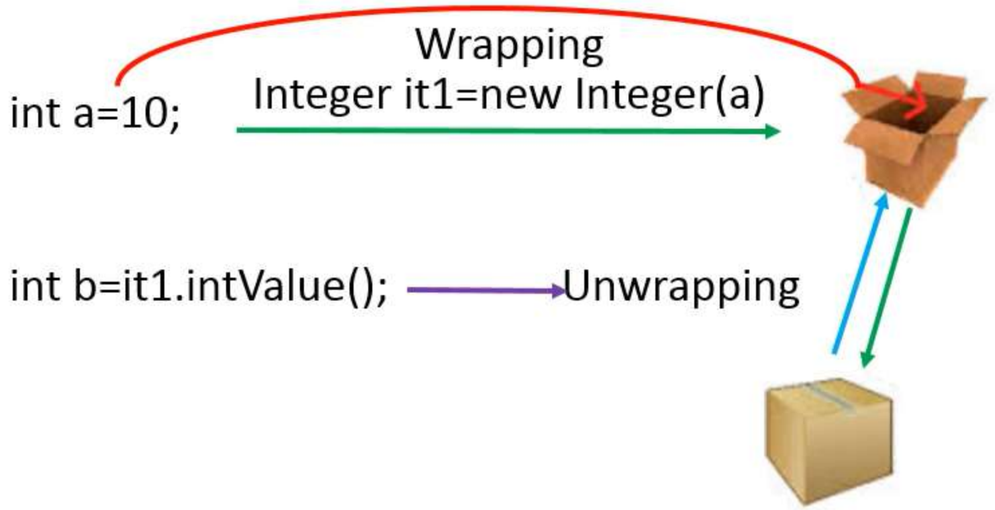
Wrapper class has methods to unwrap the object and give the data

Primitive	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Instantiating Wrapper Class

Instantiating wrapper class

- All Wrapper classes can be instantiated using their primitive type as the argument
  - Integer iobj=new Integer(10);
  - Double num=new Double(8.2);
  - Character cobj=new Character('e');
  - Boolean isdone=new Boolean(false);



Wrapping - int datatype a is converted to object it1

Use it1 when a is required as object

Unwrapping - Data is unwrapped using the method intValue() in Integer class

Type casting

TEKNOTURF

All wrapper classes have many methods that help convert from the associated type to another type

```
Integer num = new Integer(4);
float flt = num.floatValue(); //stores 4.0 in flt

Double dbl = new Double(8.2)
int val = dbl.intValue(); //stores 8 in val
```

Each numeric class contains a static method to convert a representation in a String to the associated primitive

```
int num1 = Integer.parseInt("3");
double num2 = Double.parseDouble("4.7");
```

Wrapper Class – Autoboxing & Unboxing

TEKNOTURF

Autoboxing and Unboxing features are introduced from Java 5

Autoboxing

- Converting a primitive value into an object of the corresponding wrapper class

  Example : Integer i=10;

UnBoxing

- Converting an object of a wrapper type to its corresponding primitive value

  Example :

```
Integer i=10;   // boxing
int y = i;      //  unboxing
```

Wrapper Class – autoboxing & unboxing

TEKNOTURF

Prior to java 5

```
Integer y = new Integer(567);       // wrap a data type
int x = y.intValue();               // unwrap it
x++;                                // use it
y = new Integer(x);                 // re-wrap it
```

From java 5 this is done automatically by the compiler itself by the concept of AutoBoxing and Unboxing

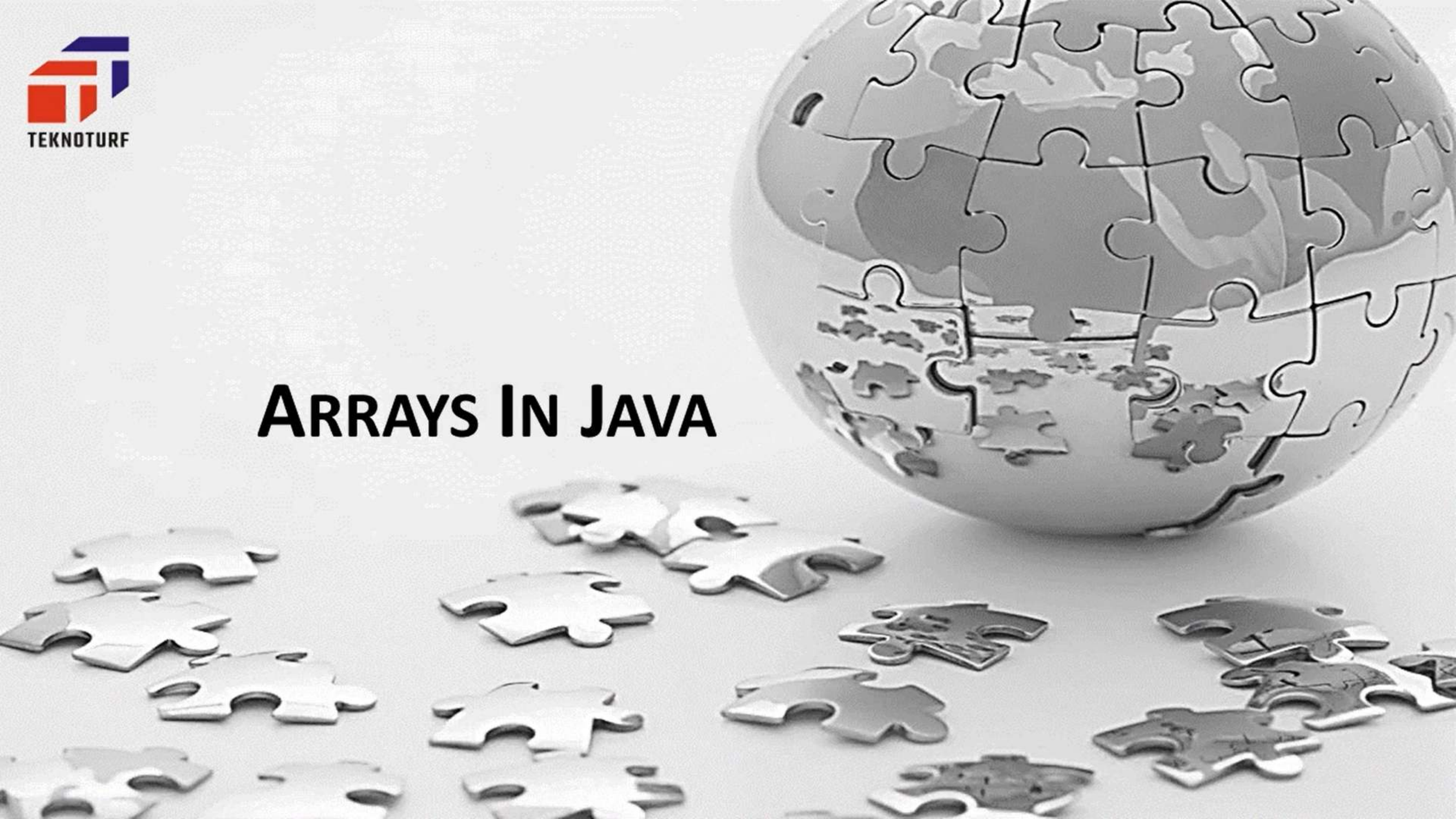
```
Integer y = new Integer(567);       // make it
  y++;                              // unwrap it, increment it, rewrap it
  System.out.println("y = " + y);   // print it
```

Summary

- Wrapper class and its usage
- Convert primitive to wrapper and vice versa
- Convert String to primitive
- Autoboxing and unboxing concept

TEKNOTURF

ARRAYS IN JAVA



Overview

TEKNOTURF

Assume Weather forecast of three states are to be stored

32 **State 1**

30 **State 2**

36 **State 3**

To store this data, we require three variables **– temperature1, temperature2, temperature 3**

Overview

TEKNOTURF

In case we need to store and manipulate the weather forecast of 10 states..

.....

32 **State 1**

30 **State 2**

36 **State 10**

To store this data, we will now require ten different variables **– temperature1, temperature2…….temperature 10.**

Overview

TEKNOTURF

Instead of 10 different variables, it is possible to store all the ten values in a single variable, say, temperature.

An array can hold multiple elements of the same data type referred by a common name.

```
int temperature[ ] = new int[10];
```

In this Module You will learn

TEKNOTURF

- Array
- Arrays classification
- One dimensional Array
- Two dimensional Array
- Array of Objects
- Usage of varargs
- Arrays class

Arrays

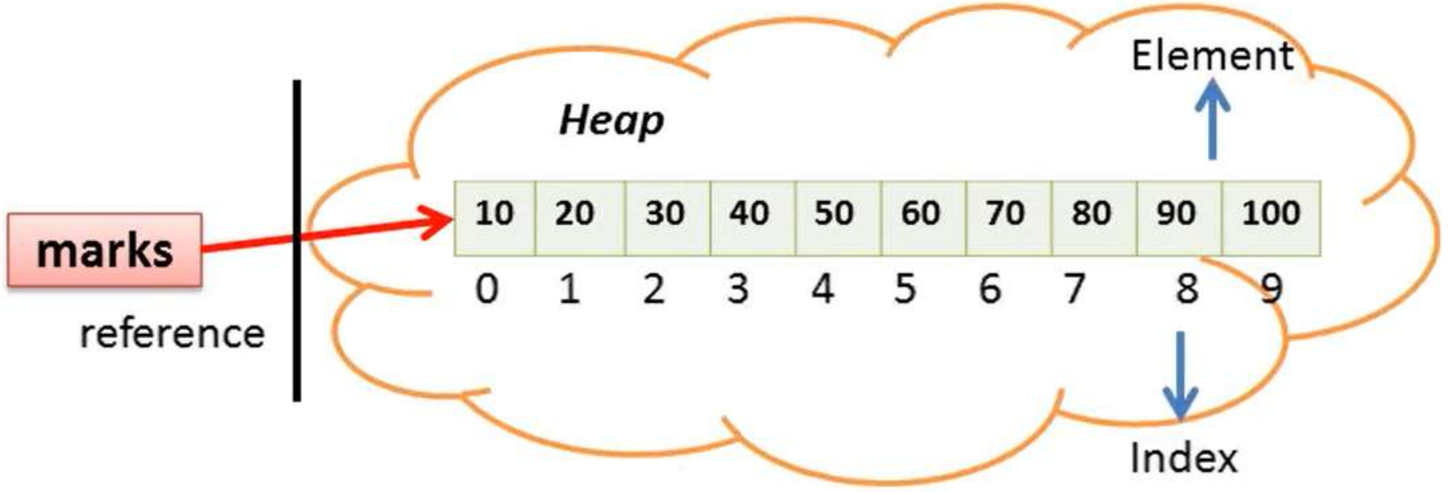
TEKNOTURF

Array is an ordered collection that stores elements of the same type.

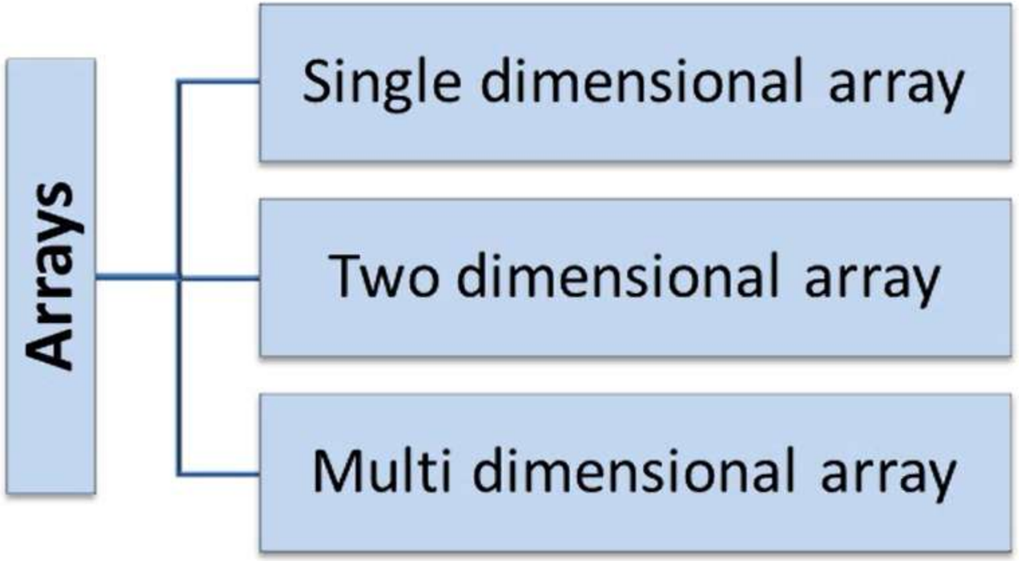
Array elements are stored contiguously.

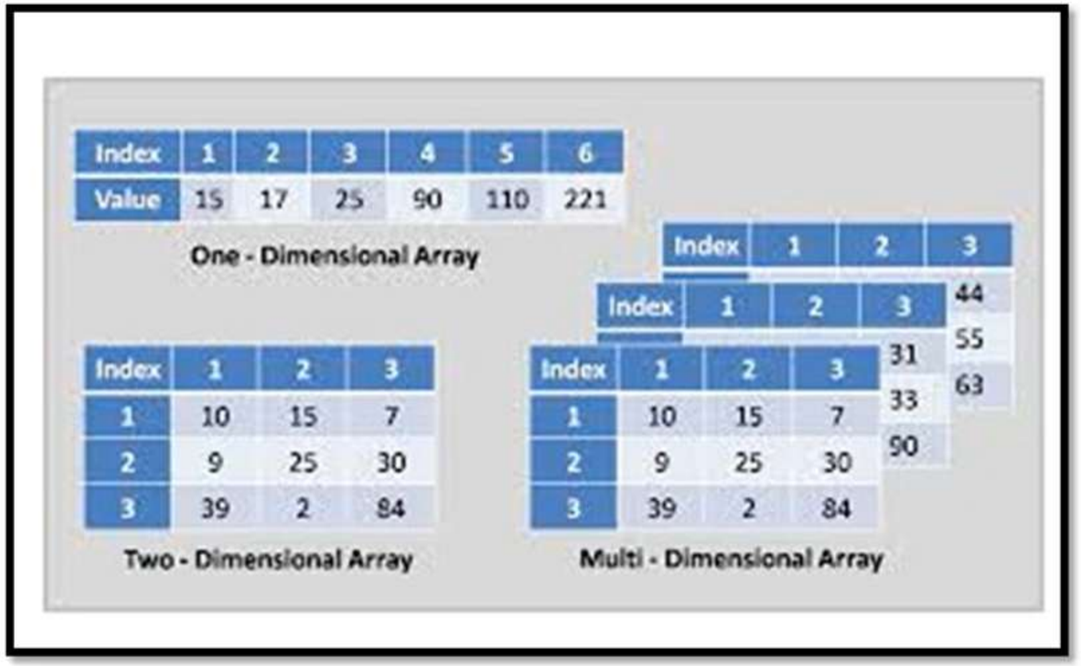
Array in java is index based; the first element of the array is stored at 0 index.

Array is a Reference data type.



Arrays Classification

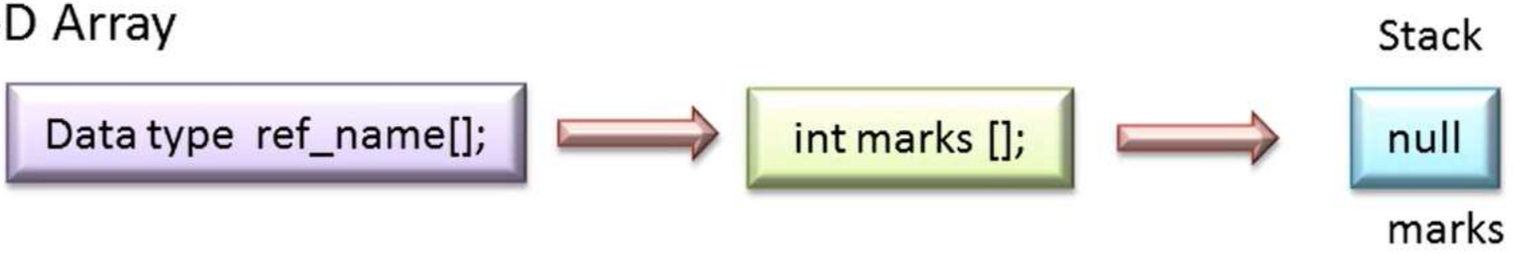




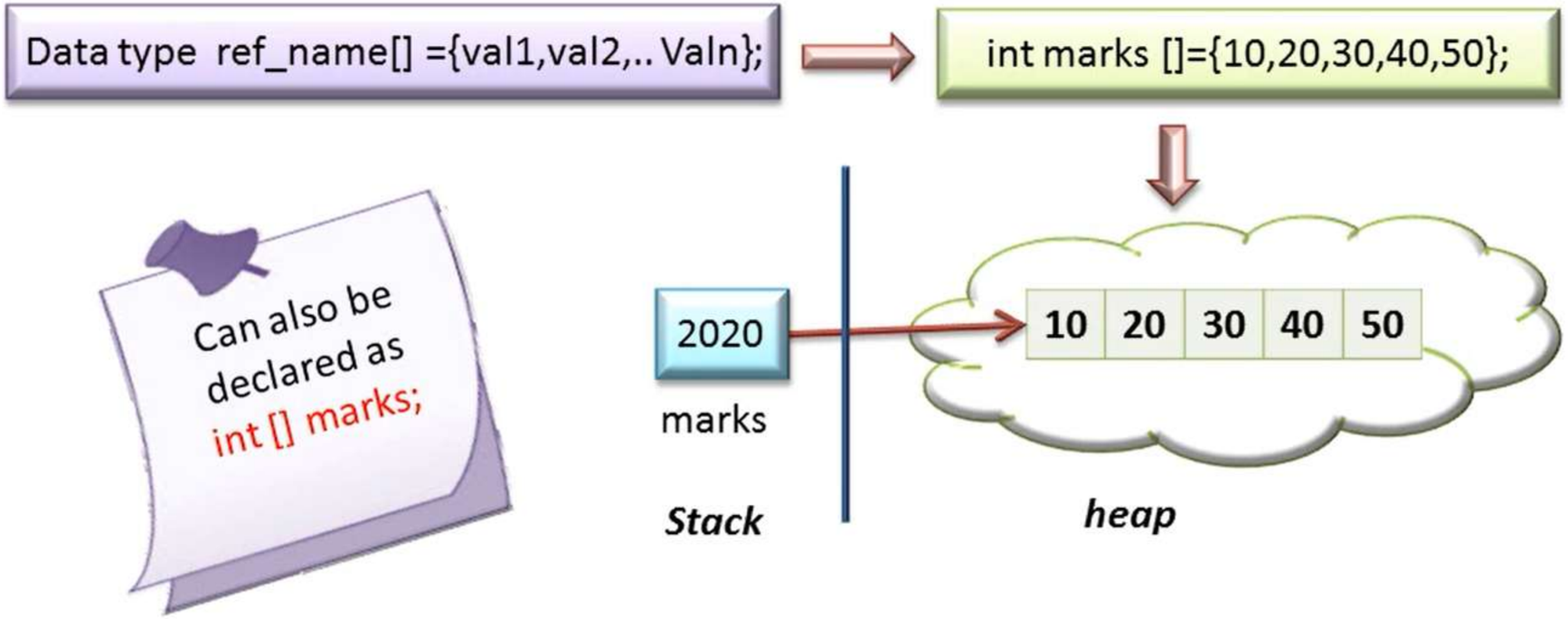
One dimensional Array

TEKNOTURF

Declaring 1-D Array



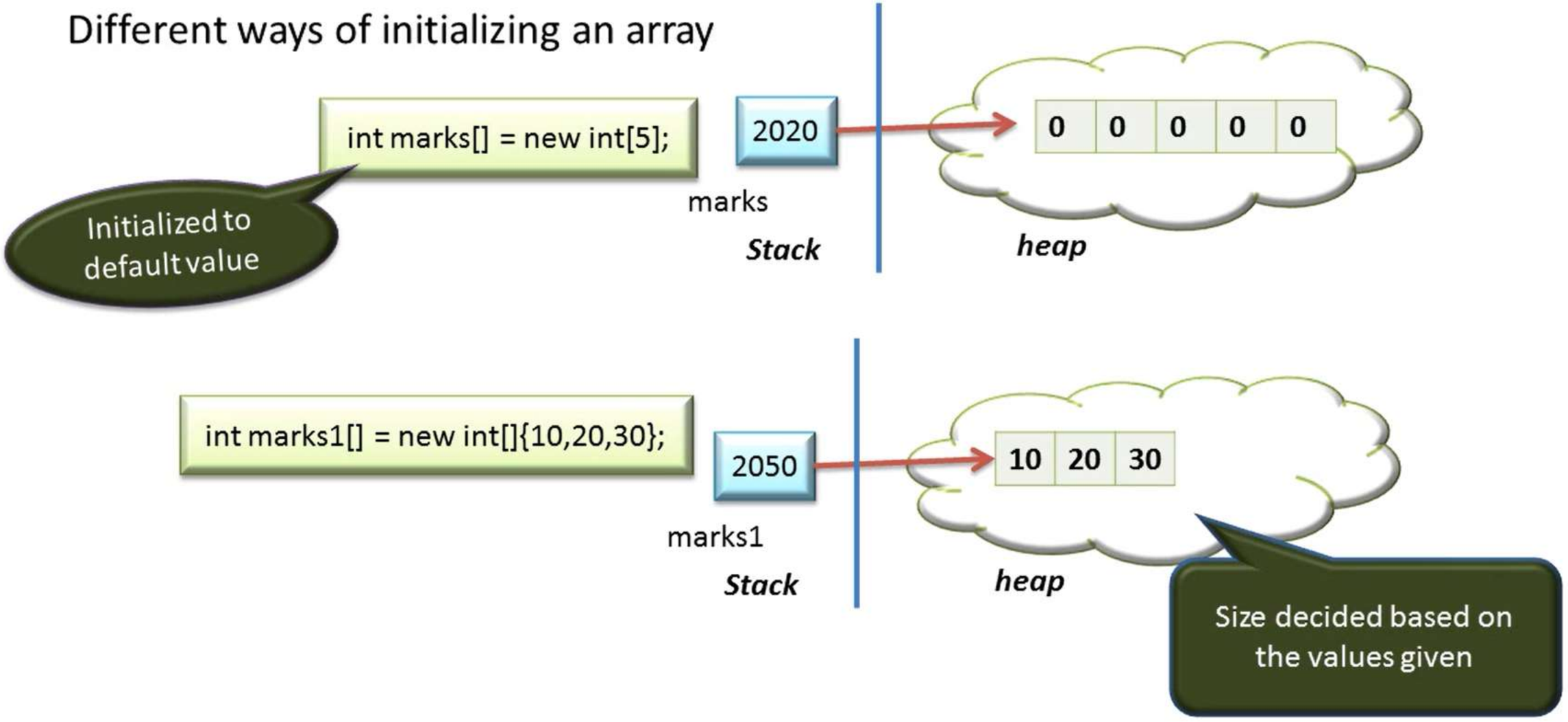
Initializing Array



One dimensional Array

TEKNOTURF

Different ways of initializing an array



One dimensional Array

TEKNOTURF

Assigning Elements in array

int marks[]=new int[3];	0	0	0
marks[0] = 10;	10	0	0
marks[1] = 20;	10	20	0
marks[2] = 30;	10	20	30

```
for(int i=0;i<3;i++){
    marks[i] = (i+1)*10;
}
```

Assigning values through for loops

One dimensional Array

TEKNOTURF

Length property

- Gets the length of the array ie. The number of elements in an array

<array_name>.length

```
int marks[]=new int[3];
for(int i=0;i<marks.length;i++){
    marks[i] = (i+1)*10;
}
```

Iterates the loop for 3 times

One dimensional Array

TEKNOTURF

Accessing elements from array

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

(OR)

Enhanced for loop introduced in Java 5

```
for (data_type variable: array_name)
```

```
for(int m : marks) {
    System.out.println(m);
}
```

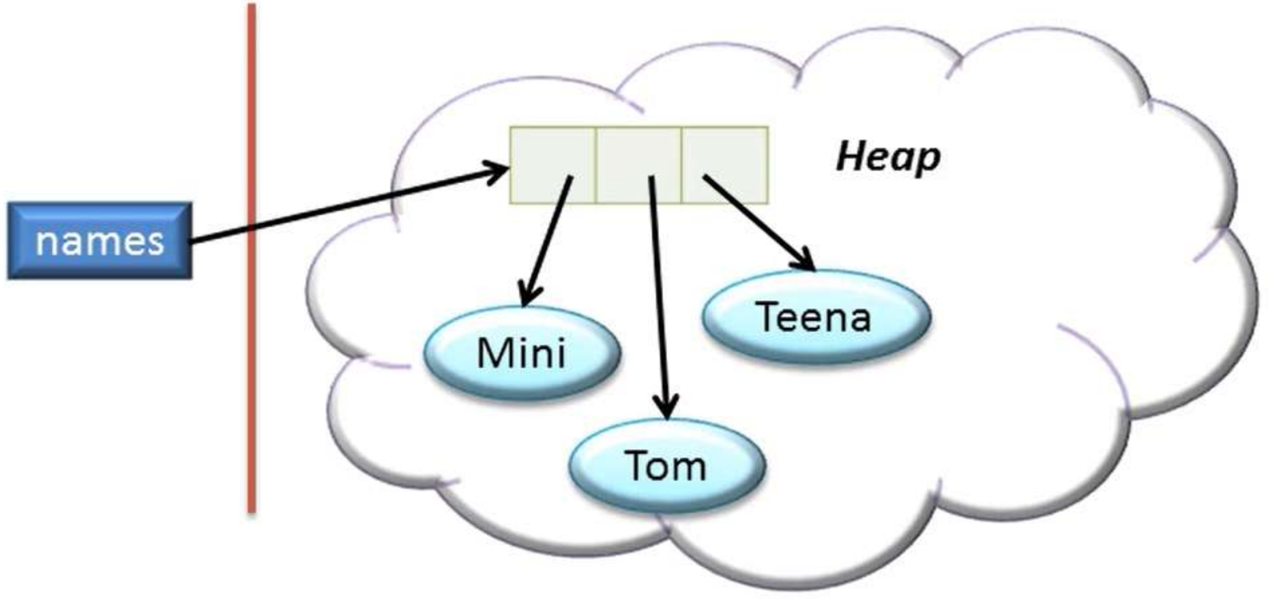
One dimensional Array

TEKNOTURF

Creating a String Array

```
String names[] = new String[3];
names[0] = new String("Mini");
names[1] = "Tom";
names[2] = new String("Teena");
```

Arrays for reference data types hold the address of the array elements



Array Of Objects

TEKNOTURF

Array can also store objects

Example

```
Employee empList[]=new Employee[3];
```

The array variable empList[] does not hold an array of Employee objects, instead it holds an array of Employee reference variables.

To make these variables hold the object, we have to create the object as

```
empList[0]=new Employee();

empList[1]=new Employee();

empList[2]=new Employee();
```

Array Of Objects - Example

TEKNOTURF

```
class Employee {
      private int empId;
      private String name;
     //Write constructor, getters and setters
}

public class ObjectArrayDemo {
      public static void main(String[] args) {
                  Employee empList[]=new  Employee[3];
                  Scanner sc=new Scanner(System.in);
                  for(int index=0;index<empList.length;index++){
                        System.out.print("Enter the id : ");
                        int id=sc.nextInt();
                        System.out.print("Enter the name : ");
                        String name=sc.next();
                        empList[index]=new  Employee(id, name);
                  }
```

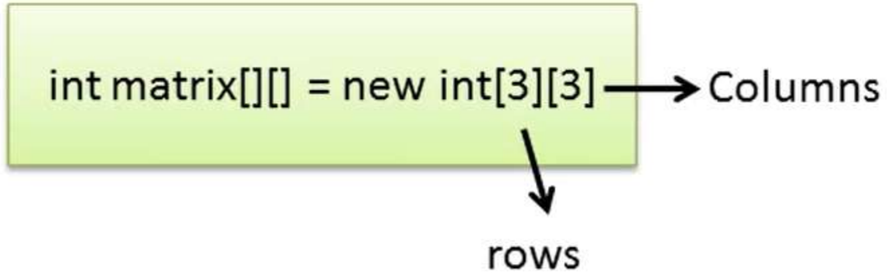
```
      for(int index=0;index<empList.length;index++){
                  System.out.println("Index is "+index);
                  System.out.println("ID is  “+
                              empList[index].getEmpId()  +
                              "Name is “ +
                              empList[index].getName());
      }
   }
}
```

Output

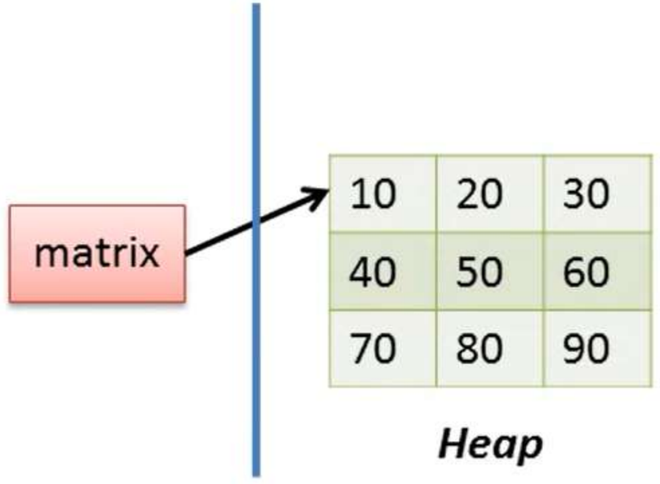
```
Enter the id : 101
Enter the name : Pooja
Enter the id : 102
Enter the name : Pinky
Enter the id : 103
Enter the name : Tina
Index is 0
ID is 101  Name is Pooja
Index is 1
ID is 102  Name is Pinky
Index is 2
ID is 103  Name is Tina
```

2-D Array

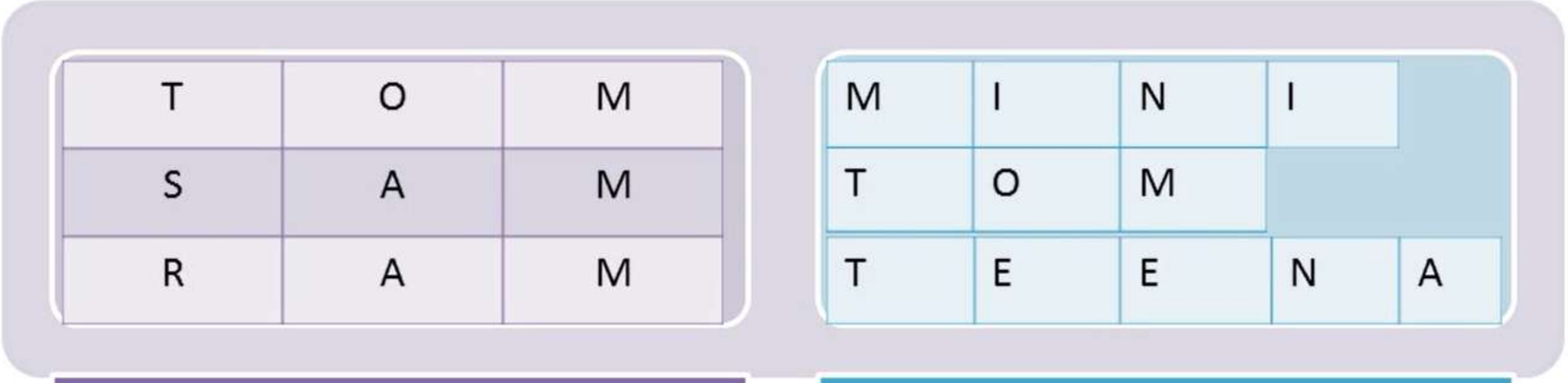
- Data is stored in the form of rows and columns
- Also known as matrix form



0	[0][0]	[0][1]	[0][2]
1	[1][0]	[1][1]	[1][2]
2	[2][0]	[2][1]	[2][2]
	0	1	2



2-D Array classification



Rectangular array

- Arrays that have elements of the same size
- char names[][] = new char[3][3];

Jagged array

- each row of the array may contain different lengths
- char names[][] = new char[3][];
  names[0] = new char[4];
  names[1] = new char[3];
  names[2] = new char[5];

2-D Array - Example

TEKNOTURF

Example - To store price list of 3 products by 2 vendors

```
int price[][]=new int[3][2];

price[0][0]=90;price[0][1]=85;
price[1][0]=180;price[1][1]=173;
price[2][0]=590;price[2][1]=490;

for(int outer=0;outer<price.length;outer++)  {
        System.out.println("Product index "+(outer+1));
        for(int inner=0;inner<price[outer].length;inner++){
                System.out.print("Vendor"+(inner+1)+
                        " - "+price[outer][inner]+"   ");
        }
        System.out.println();
}
```

Output

```
Product index 1
Vendor1 - 90  Vendor2 - 85
Product index 2
Vendor1 - 180 Vendor2 - 173
Product index 3
Vendor1 - 590 Vendor2 - 490
```

Variable Arguments

TEKNOTURF

Variable Arguments (varargs) is a newly implemented feature of Java 5.0

It allows a method to accept zero or more arguments

If the number of arguments to a method are of same type, but the number of arguments is unknown, varargs is a better approach

Syntax

return_type method_name (data_type ... variable_name) { ..... }

Variable Arguments

TEKNOTURF

```
public static void main(String[] args)
{
      System.out.println("The sum is " +
                              sum(10,20,30));
}
public int  sum (double... numbers)
{
      int total = 0;
      for (int i = 0; i < numbers.length; i++)
        total += numbers[i];
      return total;
}
```

Argument passed to a method is converted into an array of the same-typed values

sum(10,20,30) ⟺ sum(new int[] {10,20,30})

The rules for varargs

- There can be only one variable argument in the method
- Variable argument must be the last argument

Variable Arguments - Example

TEKNOTURF

```
class VarargsExample
{
        static void display(int num, String... values)  {
                System.out.println("number is "+num);
                for(String s:values){
                   System.out.println(s);
                }
        }

         public static void main(String args[]){
                display(500,"hello");//one argument
               display(1000,"my","name","is","varargs");//four arguments

        }
}
```

Output :

number is 500
hello
number is 1000
my name is varargs

Arrays

TEKNOTURF

Arrays is a class in java.util package

Arrays class has a collection of static methods to work with arrays like sorting and searching

All methods in Arrays are static and hence have no constructor.

Few methods in Arrays class are

- equals
- copyOf
- sort
- binarySearch

Arrays.equals

TEKNOTURF

boolean equals(array1, array2)

- This method compares two arrays
- Both arrays should be of same data type and one dimensional
- Return true if both arrays contain same elements in the same order

```
int  a[]={2,4,6,8,10};
int b[]={2,4,6,8,10};
System.out.println(Arrays.equals(a,b)); //returns  true

int  c[]={2,4,6,8,10};
int d[]={10,8,4,2,6};
System.out.println(Arrays.equals(a,b)); //returns  false
```

Arrays.copyOf

TEKNOTURF

array copyOf(originalArray, newLength)

- This method returns an array which is a copy of the originalArray
- It copies elements from 0 to newLength and puts them in the new array
- If newLength exceeds the length of originalArray, it pads those positions with default value.

```
int  num1[]={2,4,6,8,10};
int num2[]=Arrays.copyOf(num1,3);

for(int num : num2)
System.out.println(num);   // prints 2 4 6

Integer num3[]={1,2,3};

Integer num4[]=Arrays.copyOf(num3,5);
 for(Integer num : num4)
System.out.println(num);   // prints  1 2 3 null null
```

the index

Arrays.sort

TEKNOTURF

void sort(array)

- This method sorts the array in ascending order
- If the array is an array of objects, it sorts them in ascending order according to their natural ordering

```
int num1[]={12,41,32,16,10};
Arrays.sort(num1);

for(int num : num1)
    System.out.println(num); // prints 10 12 16 32 41

int num2[]={82,41,32,16,10,46,72,89};
Arrays.sort(num2,1,4); //sorts the numbers from 1 (inclusive) to 4 (exclusive)

for(int num : num2)
System.out.println(num); //prints 82 16 32 41 10 46 72 89
```

Arrays.binarySearch

TEKNOTURF

int binarySearch(array,key)

- This method searches for the specified key in the array.
- For this, the array should be sorted prior to this method call.
- Returns the index position where the key is found
- If key not found, returns -1
- If array not sorted, result is undefined

```
int num[]={82,41,32,16,10,46,72,89};
System.out.println(Arrays.binarySearch(num, 41));  // output unpredictable

Arrays.sort(num);
System.out.println(Arrays.binarySearch(num, 41)); // prints 3
```

Summary

- Array
- Arrays classification
- One dimensional Array
- Two dimensional Array
- Array of Objects
- Usage of varargs
- Arrays class

TEKNOTURF

STRING IN JAVA

In this Module You will learn

TEKNOTURF

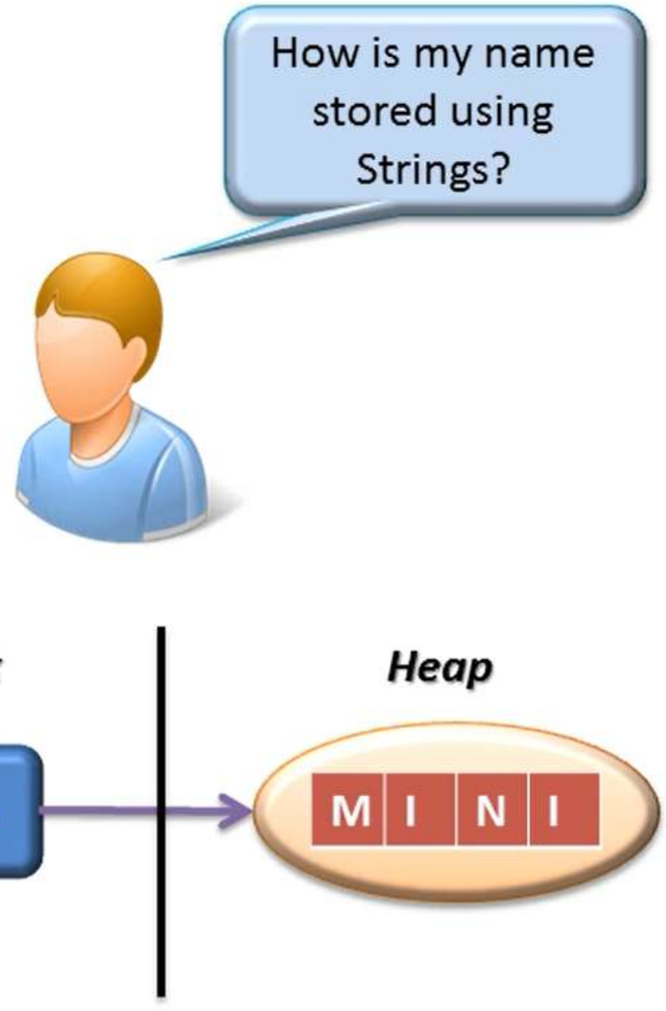
- String class
- String Functions
- String Buffer and String Builder Class

String

String is a Reference data type and is used to store sequence of characters

String is a class in java.lang package

String API contains many built-in operations like compare, concat, equals, split, length, replace, compareTo, substring etc.



TEKNOTURF

Creation of String

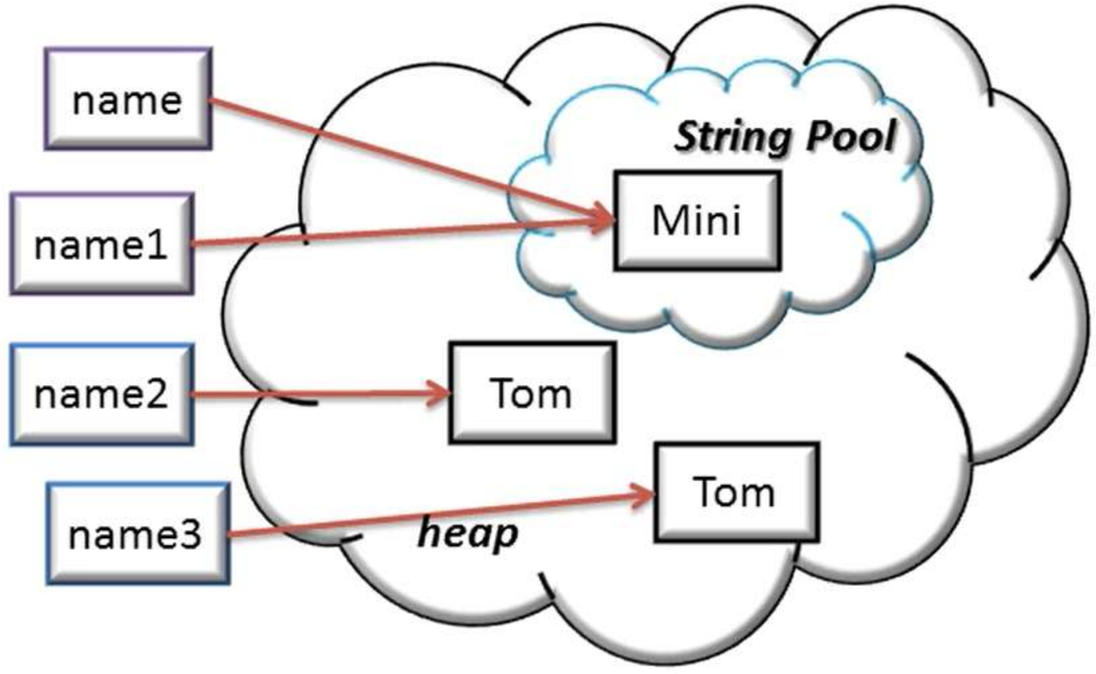
TEKNOTURF

Created in string pool

- String name="Mini";
- String name1="Mini";

By new Keyword

- String name2=new String("Tom");
- String name3=new String("Tom");



String Pool makes Java more memory efficient. No new Objects are created if a similar object already exists

String

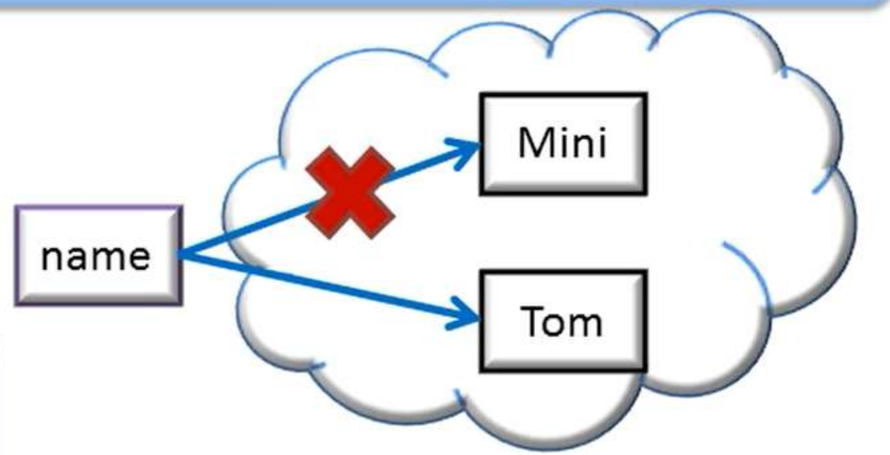
String Objects are immutable

Immutable means something that cannot be modified or changed.

Once string object is created, its data or state can't be changed but a new string object is created.

String name="Mini";

name="Tom"



The Object "Mini" is unmodified. Instead a new Object "Tom" is created and mapped with name reference.

String Functions

TEKNOTURF

String API contains many built in methods. Some of the methods are

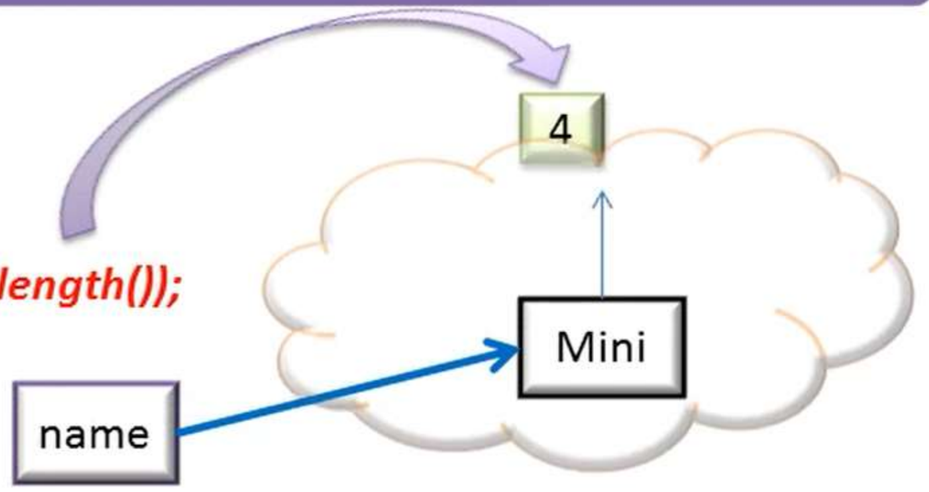
Function	Function Signature	Purpose
length	int length()	returns string length
concat	String concat(String str)	concatenates specified string
charAt	char charAt(int index)	returns char value for the particular index
equals	boolean equals(Object another)	checks the equality of string with another String object
substring	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index

String Function

TEKNOTURF

Length

```
String name="Mini";
System.out.println("length: " + name.length());
```

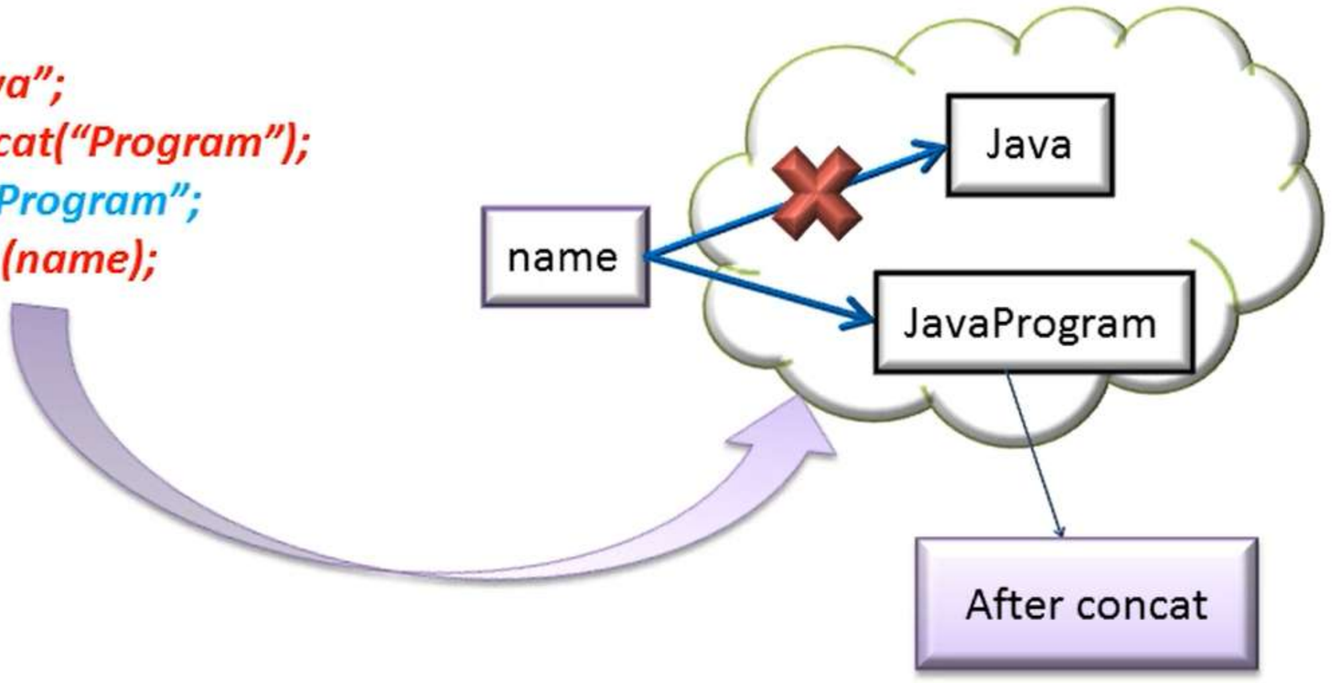


String Function

TEKNOTURF

Concat

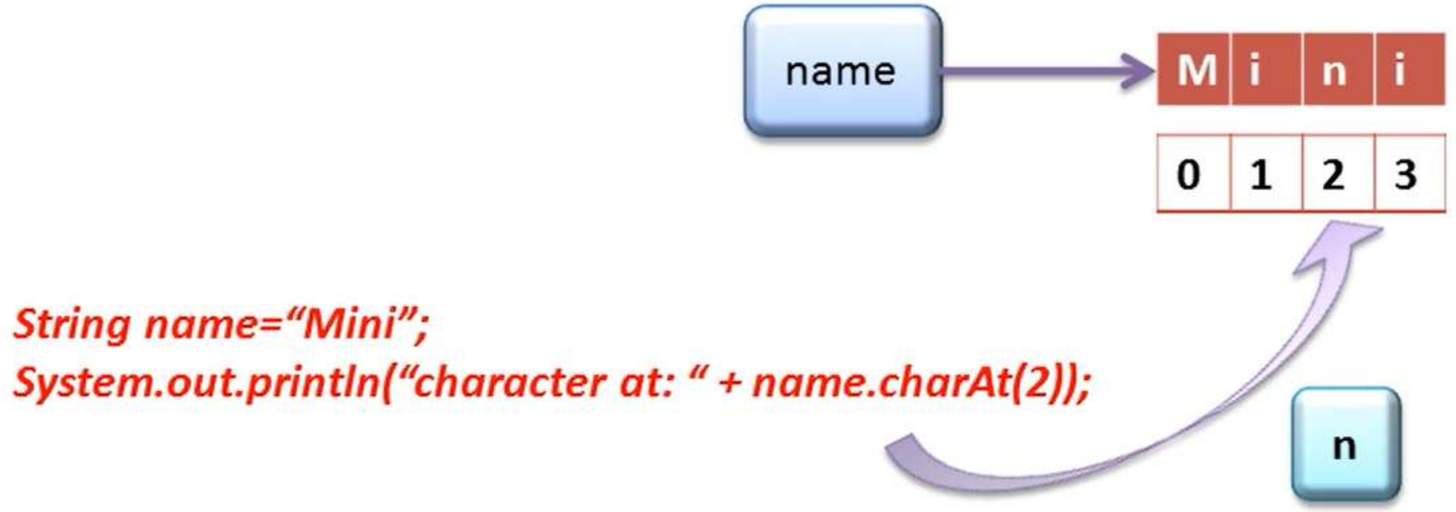
```
String name = "Java";
name = name.concat("Program");
//name = name+"Program";
System.out.println(name);
```



String Function

TEKNOTURF

charAt




```
String name="Mini";
System.out.println("character at: " + name.charAt(2));
```

String Function

TEKNOTURF

equals

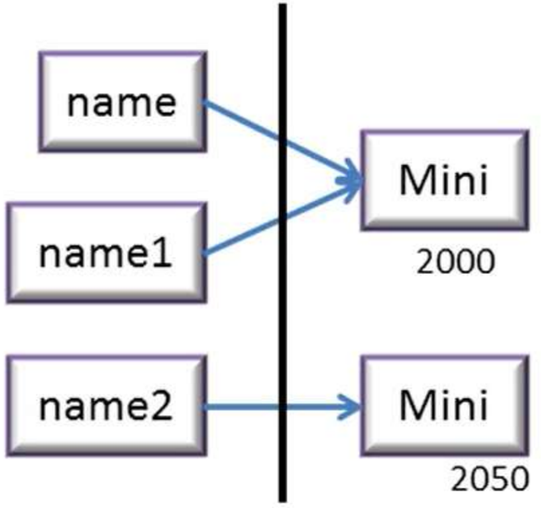
equalsIgnoreCase – case insensitive check

```
String name="Mini";
String name1="Mini";
String name2=new String("Mini");
System.out.println(name == name1);
System.out.println(name == name2);
System.out.println(name.equals(name2));
```

→ true

→ false

→ true



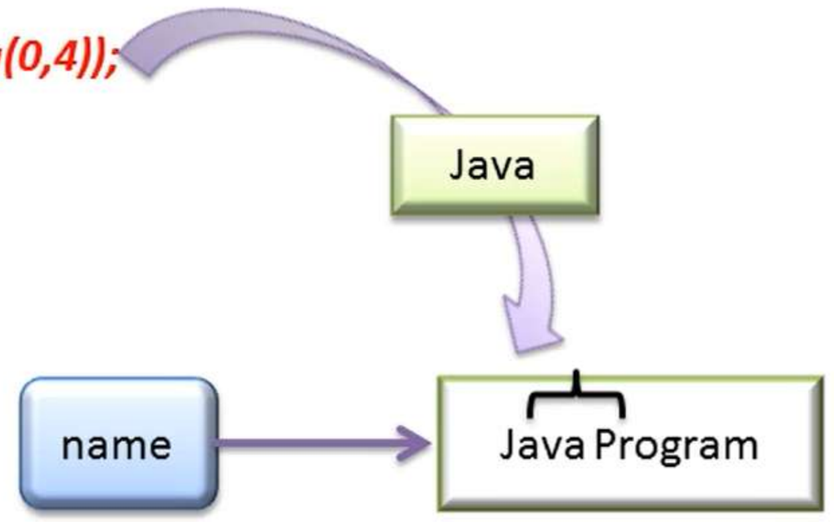
== checks for the address of the string Objects to be compared

equals checks for the values inside the objects to be compared

String Function

substring

```
String name="Java Program";
System.out.println(name.substring(0,4));
```

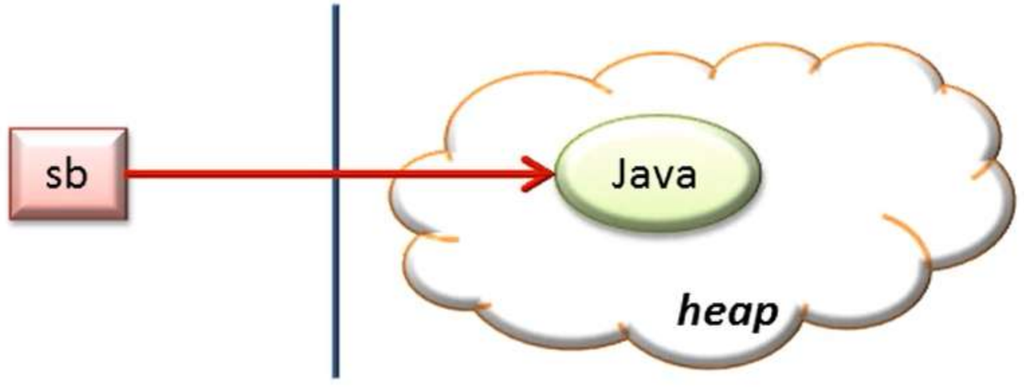


Starting from 0, 4 characters long is the substring data

StringBuffer

- StringBuffer is used to create a mutable String
- StringBuffer class is thread-safe
- Creating a StringBuffer Object

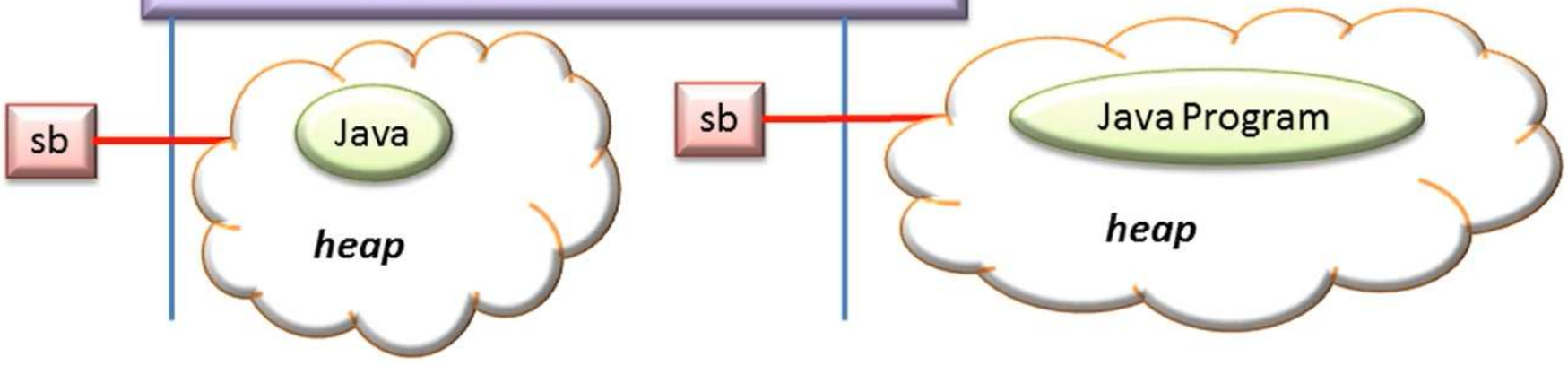
```
StringBuffer sb = new StringBuffer("Java");
```



StringBuffer Function

- append
  - StringBuffer append(String s)
  - Appends the specified string with given string.


```
StringBuffer sb = new StringBuffer("Java");
sb.append(" Program");
System.out.println(sb);
```



StringBuffer Function

TEKNOTURF

```
StringBuffer sb=new StringBuffer("Java");
```

Function signature	Purpose	Example
char charAt(int index)	returns the character at the specified position.	System.out.println(sb.charAt(2)); // v
int length()	returns the length of the string	System.out.println(sb.length()); // 4
String substring(int beginIndex, int endIndex)	returns the substring from the specified beginIndex and endIndex.	String s = sb.substring(0,4); System.out.println(s); // Java

StringBuffer reverse()	used to reverse the string	sb.reverse();
		System.out.println(sb);
		// avaJ

StringBuffer to String

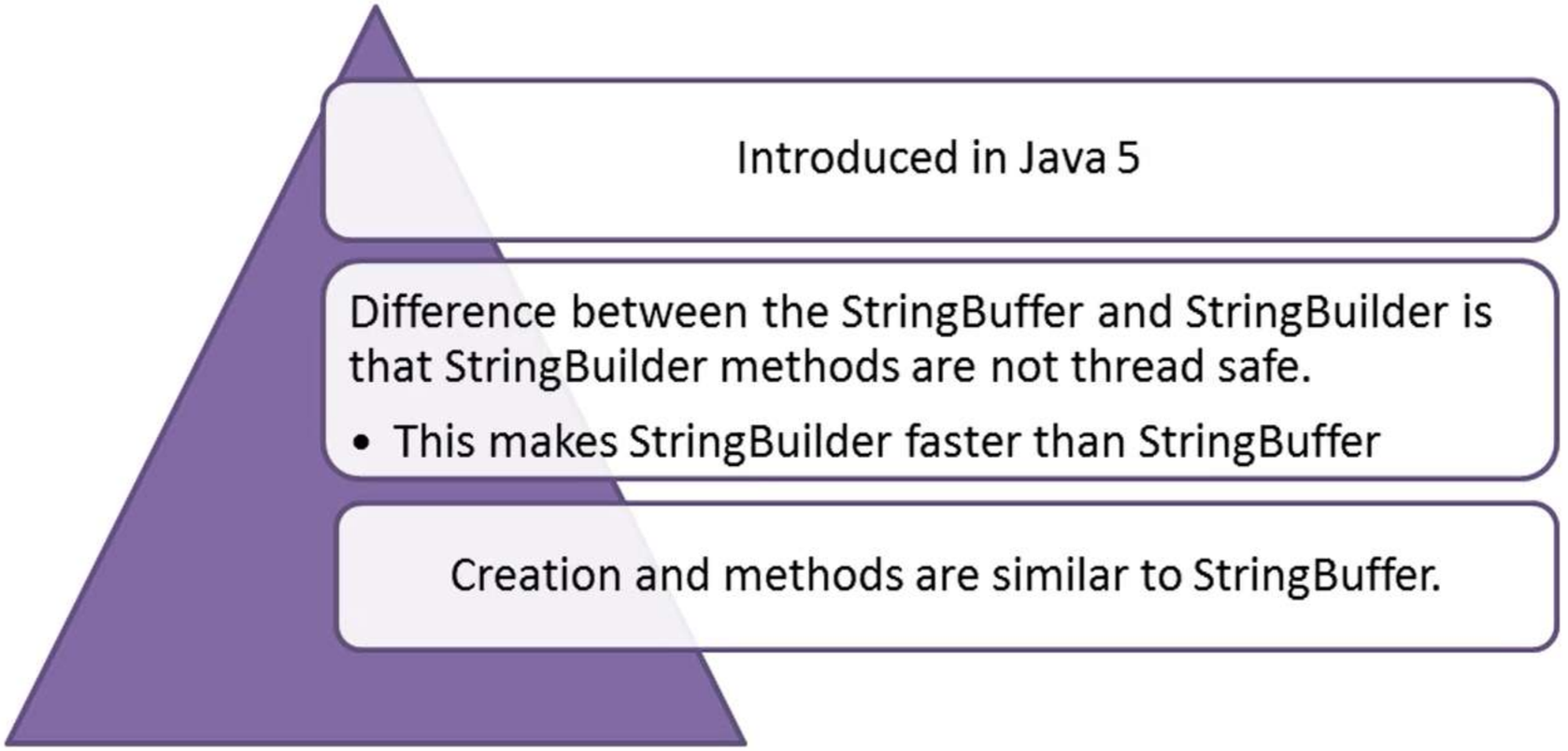
TEKNOTURF

- StringBuffer object can be converted to String using toString()
- **toString functions return the String representation of reference Object**
- toString method is present in the Object class.

```
StringBuffer name= new StringBuffer("Raghu");
String sname=name.toString();
System.out.println(sname.equals("Raghu"));
```

To check whether the given StringBuffer Object's value is equal to Raghu, we need to convert StringBuffer to String and use equals method

StringBuilder



Summary

- String class
- String Functions
- String Buffer and String Builder Class

Java - ArraysAndString/src/ArrayDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- ABCCollege
- ArraysAndString
  - src
    - (default package)
      - ArrayDemo.java
      - FindMaxInArray.java
  - JRE System Library [JavaSE-1.7]
- JavaIntro

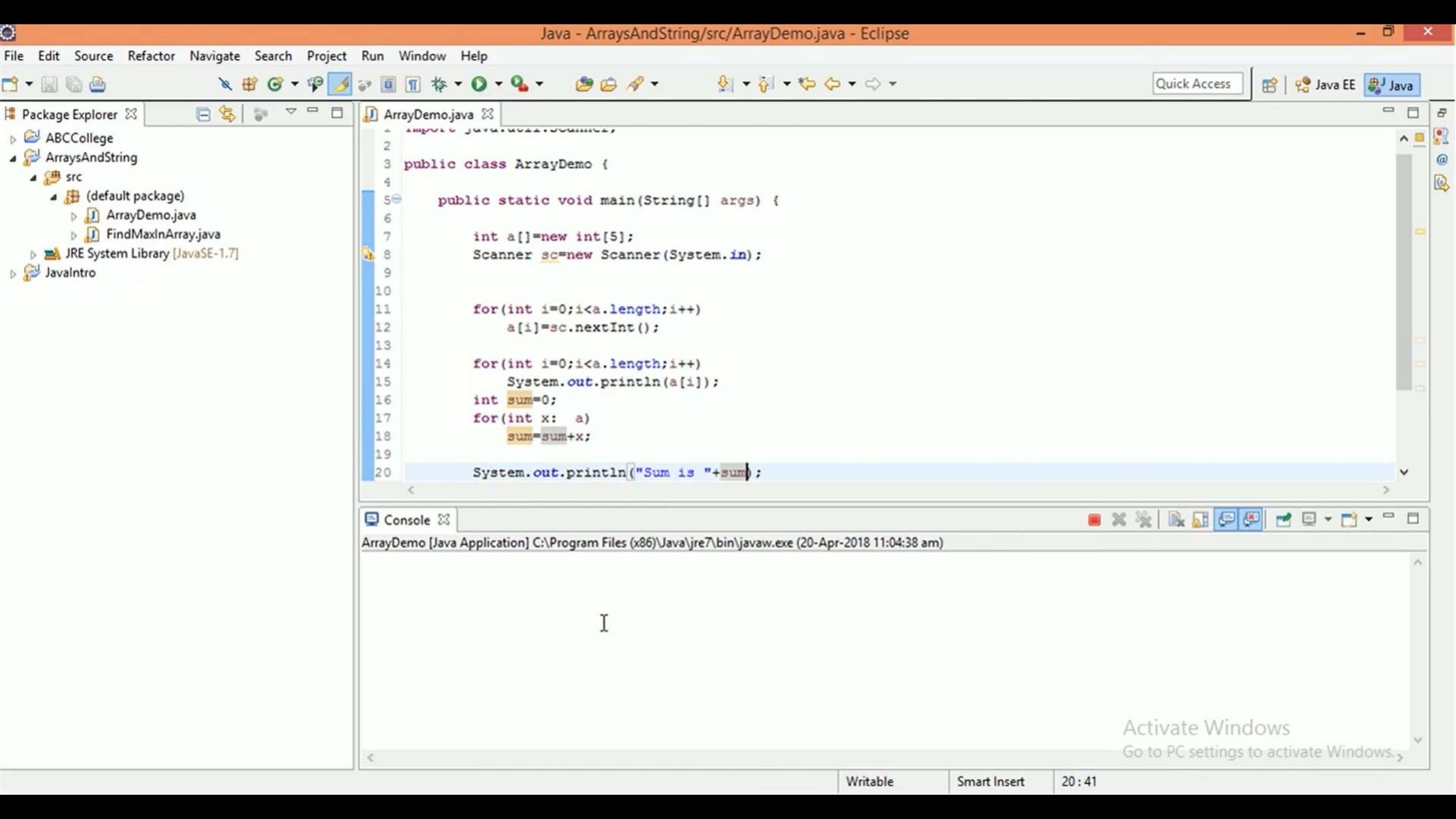
*ArrayDemo.java

```
2
3  public class ArrayDemo {
4
5      public static void main(String[] args) {
6
7          int a[]=new int[5];
8          Scanner sc=new Scanner(System.in);
9
10
11         for(int i=0;i<a.length;i++)
12             a[i]=sc.nextInt();
13
14         for(int i=0;i<a.length;i++)
15             System.out.println(a[i]);
16         int sum=0;
17         for(int x:  a)
18             sum=sum+x;
19
20
```

Console

No consoles to display at this time.

Writable | Smart Insert | 20 : 9


Java - ArraysAndString/src/ArrayDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
ABCCollege
ArraysAndString
src
(default package)
ArrayDemo.java
FindMaxInArray.java
JRE System Library [JavaSE-1.7]
JavaIntro
ArrayDemo.java
2
3 public class ArrayDemo {
4
5 public static void main(String[] args) {
6
7 int a[]=new int[5];
8 Scanner sc=new Scanner(System.in);
9
10
11 for(int i=0;i<a.length;i++)
12 a[i]=sc.nextInt();
13
14 for(int i=0;i<a.length;i++)
15 System.out.println(a[i]);
16 int sum=0;
17 for(int x: a)
18 sum=sum+x;
19
20 System.out.println("Sum is "+sum);
Console
ArrayDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:04:38 am)
Activate Windows
Go to PC settings to activate Windows.
Writable
Smart Insert
20 : 41

Java - ArraysAndString/src/ArrayDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- ABCCollege
- ArraysAndString
  - src
    - (default package)
      - ArrayDemo.java
      - FindMaxInArray.java
  - JRE System Library [JavaSE-1.7]
- JavaIntro

ArrayDemo.java

```
1  import java.util.Scanner;
2
3  public class ArrayDemo {
4
5      public static void main(String[] args) {
6
7          int a[]=new int[5];
8          Scanner sc=new Scanner(System.in);
9
10
11         for(int i=0;i<a.length;i++)
12             a[i]=sc.nextInt();
13
14         for(int i=0;i<a.length;i++)
15             System.out.println(a[i]);
16         int sum=0;
17         for(int x:  a)
18             sum=sum+x;
19
20         System.out.println("Sum is "+sum);
```

Console

ArrayDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:04:38 am)

```
1
2
3
4
5
```

Java - ArraysAndString/src/ArrayDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- ABCCollege
- ArraysAndString
  - src
    - (default package)
      - ArrayDemo.java
      - FindMaxInArray.java
  - JRE System Library [JavaSE-1.7]
- JavaIntro

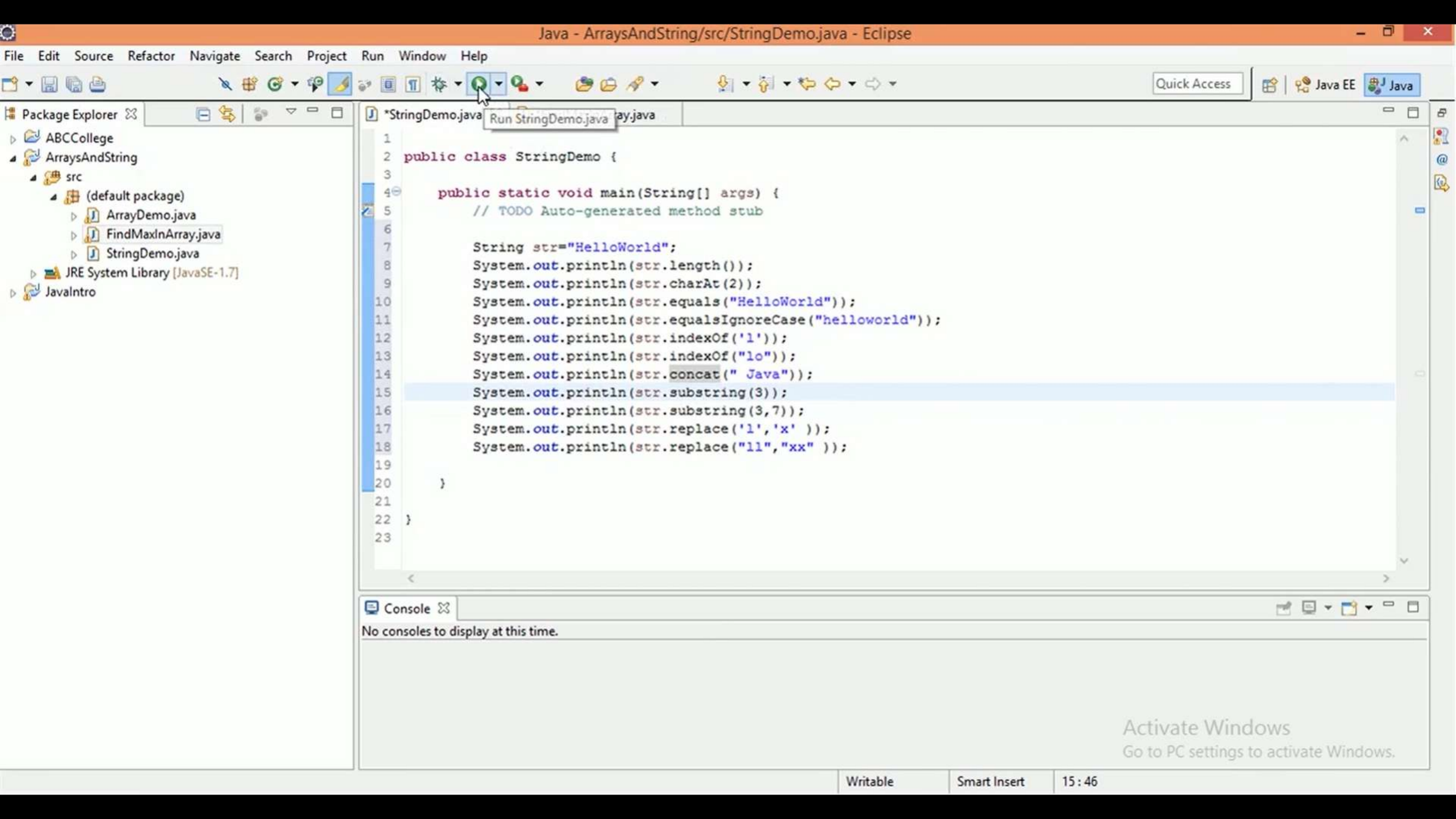
ArrayDemo.java

```
2
3  public class ArrayDemo {
4
5      public static void main(String[] args) {
6
7          int a[]=new int[5];
8          Scanner sc=new Scanner(System.in);
9
10
11         for(int i=0;i<a.length;i++)
12             a[i]=sc.nextInt();
13
14         for(int i=0;i<a.length;i++)
15             System.out.println(a[i]);
16         int sum=0;
17         for(int x:  a)
18             sum=sum+x;
19
20         System.out.println("Sum is "+sum);
```

Console

<terminated> ArrayDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:04:38 am)

```
3
4
5
1
2
3
4
5
Sum is 15
```


Java - ArraysAndString/src/StringDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
ABCCollege
ArraysAndString
src
(default package)
ArrayDemo.java
FindMaxInArray.java
StringDemo.java
JRE System Library [JavaSE-1.7]
JavaIntro
*StringDemo.java
Run StringDemo.java
public class StringDemo {
public static void main(String[] args) {
// TODO Auto-generated method stub
String str="HelloWorld";
System.out.println(str.length());
System.out.println(str.charAt(2));
System.out.println(str.equals("HelloWorld"));
System.out.println(str.equalsIgnoreCase("helloworld"));
System.out.println(str.indexOf('l'));
System.out.println(str.indexOf("lo"));
System.out.println(str.concat(" Java"));
System.out.println(str.substring(3));
System.out.println(str.substring(3,7));
System.out.println(str.replace('l','x'));
System.out.println(str.replace("ll","xx"));
}
}
Console
No consoles to display at this time.
Activate Windows
Go to PC settings to activate Windows.
Writable
Smart Insert
15 : 46

Java - ArraysAndString/src/StringDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- ABCCollege
- ArraysAndString
  - src
    - (default package)
      - ArrayDemo.java
      - FindMaxInArray.java
      - StringDemo.java
  - JRE System Library [JavaSE-1.7]
- JavaIntro

StringDemo.java | FindMaxInArray.java

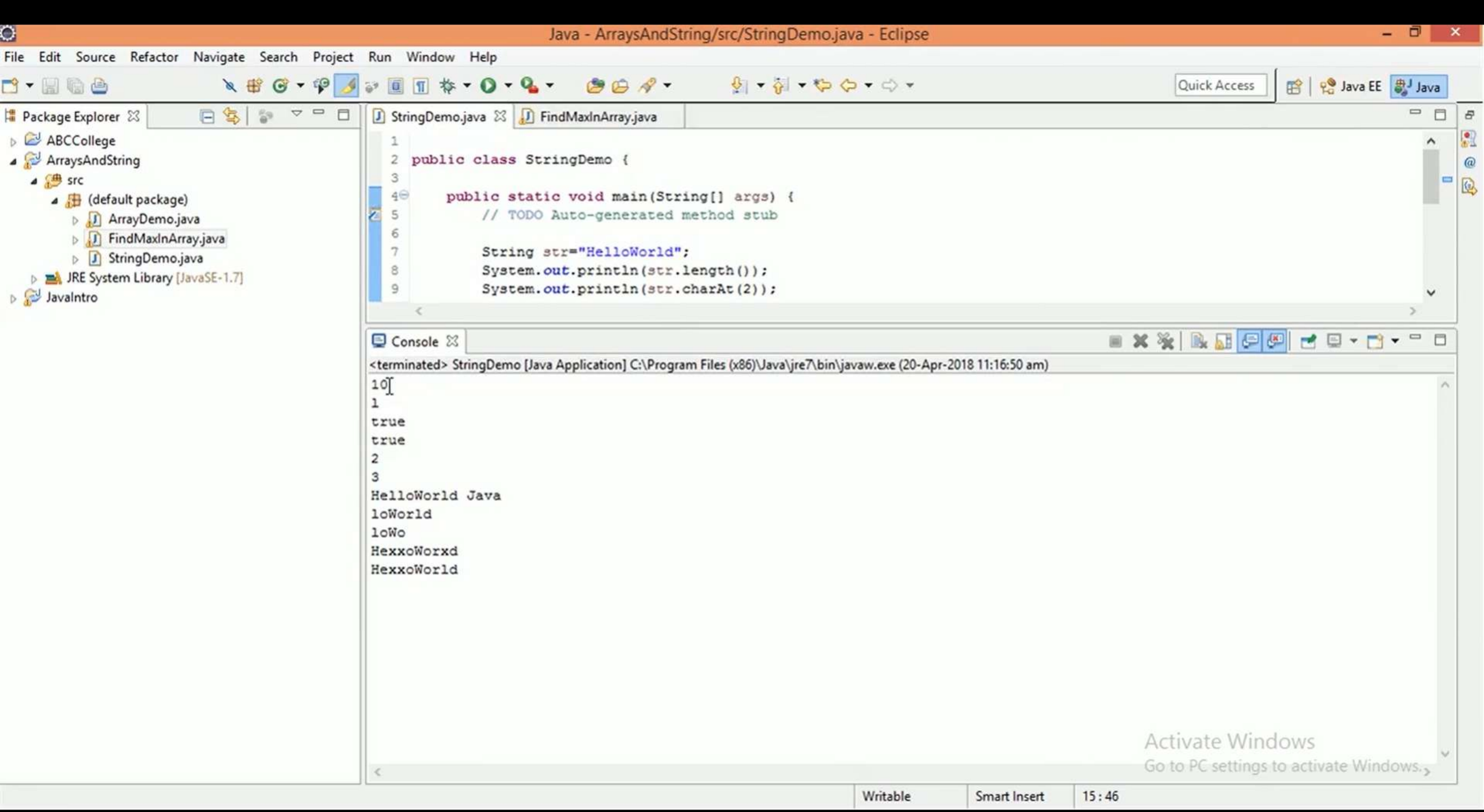
```
1
2  public class StringDemo {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          String str="HelloWorld";
8          System.out.println(str.length());
9          System.out.println(str.charAt(2));
10         System.out.println(str.equals("HelloWorld"));
11         System.out.println(str.equalsIgnoreCase("helloworld"));
12         System.out.println(str.indexOf('l'));
13         System.out.println(str.indexOf("lo"));
14         System.out.println(str.concat(" Java"));
15         System.out.println(str.substring(3));
16         System.out.println(str.substring(3,7));
17         System.out.println(str.replace('l','x' ));
18         System.out.println(str.replace("ll","xx" ));
19
20     }
21
22 }
23
```

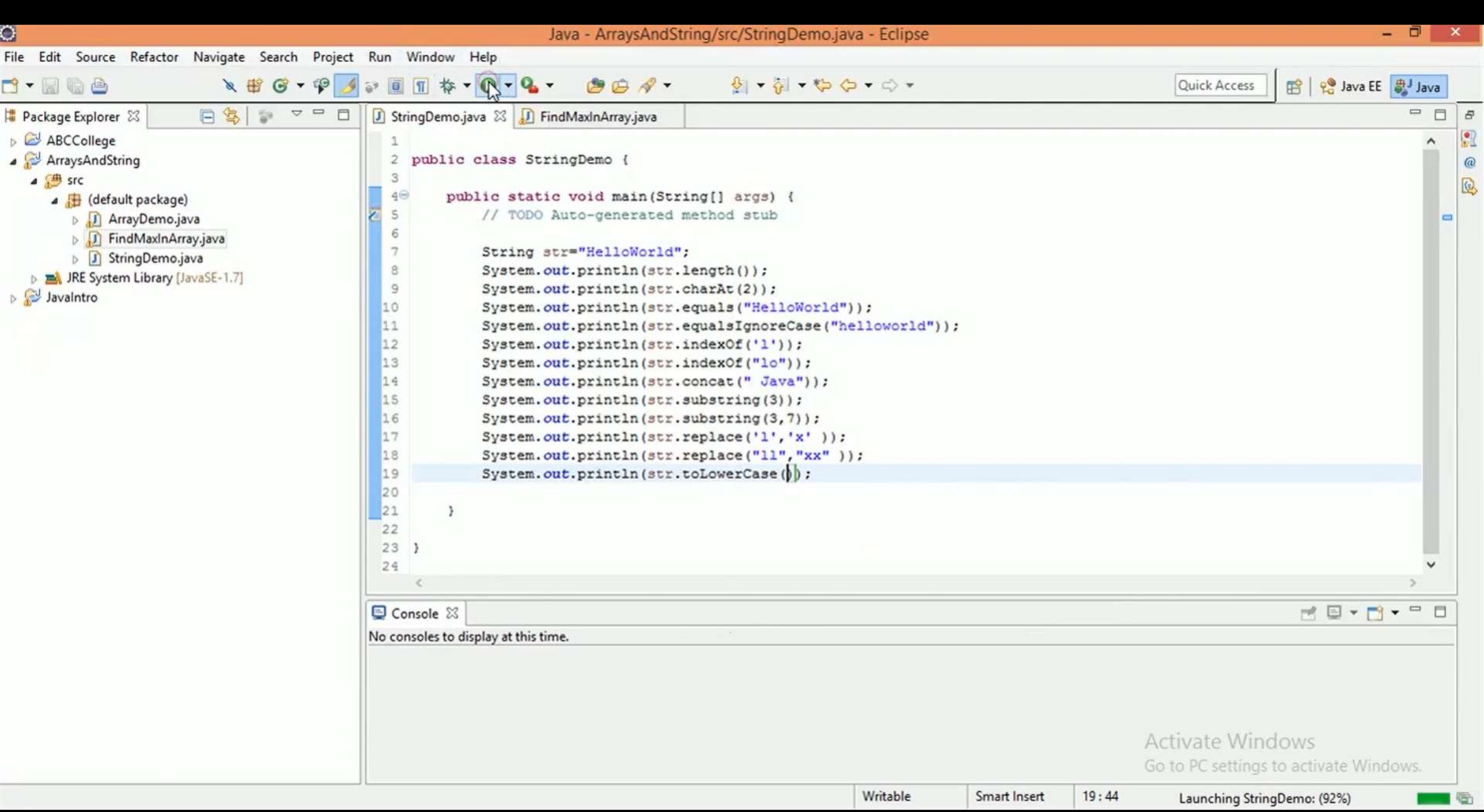
Console

<terminated> StringDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:16:50 am)

```
loWorld
loWo
HexxoWorxd
HexxoWorld
```


Writable | Smart Insert | 15 : 46


Java - ArraysAndString/src/StringDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
ABCCollege
ArraysAndString
src
(default package)
ArrayDemo.java
FindMaxInArray.java
StringDemo.java
JRE System Library [JavaSE-1.7]
JavaIntro
StringDemo.java
FindMaxInArray.java
1
2 public class StringDemo {
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6
7 String str="HelloWorld";
8 System.out.println(str.length());
9 System.out.println(str.charAt(2));
Console
<terminated> StringDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:16:50 am)
10
1
true
true
2
3
HelloWorld Java
loWorld
loWo
HexxoWorxd
HexxoWorld
Activate Windows
Go to PC settings to activate Windows.
Writable
Smart Insert
15 : 46


Java - ArraysAndString/src/StringDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Package Explorer
ABCCollege
ArraysAndString
src
(default package)
ArrayDemo.java
FindMaxInArray.java
StringDemo.java
JRE System Library [JavaSE-1.7]
JavaIntro
StringDemo.java
FindMaxInArray.java
1
2 public class StringDemo {
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6
7 String str="HelloWorld";
8 System.out.println(str.length());
9 System.out.println(str.charAt(2));
10 System.out.println(str.equals("HelloWorld"));
11 System.out.println(str.equalsIgnoreCase("helloworld"));
12 System.out.println(str.indexOf('l'));
13 System.out.println(str.indexOf("lo"));
14 System.out.println(str.concat(" Java"));
15 System.out.println(str.substring(3));
16 System.out.println(str.substring(3,7));
17 System.out.println(str.replace('l','x'));
18 System.out.println(str.replace("ll","xx"));
19 System.out.println(str.toLowerCase());
20
21 }
22
23 }
24
Console
No consoles to display at this time.
Activate Windows
Go to PC settings to activate Windows.
Writable
Smart Insert
19 : 44
Launching StringDemo: (92%)

Java - ArraysAndString/src/StringDemo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java EE | Java

Package Explorer

- ABCCollege
- ArraysAndString
  - src
    - (default package)
      - ArrayDemo.java
      - FindMaxInArray.java
      - StringDemo.java
  - JRE System Library [JavaSE-1.7]
- JavaIntro

StringDemo.java | FindMaxInArray.java

```
1
2  public class StringDemo {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          String str="HelloWorld";
8          System.out.println(str.length());
9          System.out.println(str.charAt(2));
10         System.out.println(str.equals("HelloWorld"));
11         System.out.println(str.equalsIgnoreCase("helloworld"));
12         System.out.println(str.indexOf('l'));
13         System.out.println(str.indexOf("lo"));
14         System.out.println(str.concat(" Java"));
15         System.out.println(str.substring(3));
16         System.out.println(str.substring(3,7));
17         System.out.println(str.replace('l','x' ));
18         System.out.println(str.replace("ll","xx" ));
19         System.out.println(str.toLowerCase());
20
21     }
22
23 }
24
```

Console

<terminated> StringDemo [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (20-Apr-2018 11:18:30 am)

```
loWorld
loWo
HexxoWorxd
HexxoWorld
helloworld
```