PROGRAMMING FUNDAMENTALS

1. CHARACTER SET:

- A character set is a set of textual and graphic symbols.
- Java uses Unicode character set.
- It defines standardized, universal character set used for representing characters and symbols as integers.
- It uses 16-bits i.e. it can represent more than 65000 unique characters.

2. Java tokens:

- It is the smallest element of program.
- It includes: keywords, identifiers, literals, punctuators, operators etc.

3. Keywords:

- There are 50 reserve keywords in java programming language.

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

4. Identifier:

- These are the variables used in java programming.
- An identifier is a name of fundamental building blocks of a program such as class, object, interface etc.

Valid Identifier	Explanation
Student	Capital letter with small letters are allowed.
student	Small letters are allowed.
student1	Digits and underscore are allowed with alphabets
INT	Keywords are allowed but we have to change case of any letter or complete word.
_NUM	Underscore at the first position is allowed.
Sum_of_ numbers	We can concatenate multiple words with underscore.
void	Keyword name cannot be given to identifier.
num‘1	Special characters are not allowed.
x y	Spaces are not allowed.
2x	Digits are allowed but not as first character

Rules to define a java identifier:

- Identifiers can contain alphabets, digits, underscore or dollar sign character.
- They must not begin with a digit.
- It contains upper case and lower case.
- They cannot be a keyword, Boolean letters or null character.

5. Literals:

- Literals are those data items whose value does not change during the program execution.
- These are also known as constants.

The type of literals in java is:

- Integer literals
- Character literals
- Floating point literals
- Boolean literals
- String literals
- Null literals

a. Integer literals:

These primary literals used in java are divided into 3 types:

- Decimal integer literals:- 0 to 9
- Hexadecimal integer literals:- 0 to 9, A to F.
- Octal integer literals:-0 to 7

b. Character literals:

- It represents a single Unicode character and appear within a pair of single quotations marks.
- Some characters are not readily printable through a keyboard such as backspace, tabs, etc. These type of characters are represented by using escape sequences (\).

Escape Character	Meaning
\n	New line
\t	Tab
\b	Backspace
\r	Carriage return
\f	Form feed
\\	Backslash
\'	Single quotation
\"	Double quotation
\ud	Unicode character
\d	Octal character
\xd	Hexadecimal character

Where the letter d such as in octal, hexa etc , represent a number.

c. **Floating point literals:**

Floating point literals can be donated as a decimal point, an fraction part and an exponent.

Part	Is It Required?	Examples
Whole number	Not, if fractional part is present	0,5,100,1005
Decimal Point	Not, if exponent is present; must be there if there is a fractional part.	e 23, 6.5E32, 6.5
Fractional	Cannot be present if there is no decimal point; must be there if there is no whole number part.	0, 3, 1415927, 0.5, 1.6, 7.5
Exponent	Only if there is no decimal point.	2 43, E-17, E5, e+304, e-2
Type suffix	No. The number is assumed to be double precision in the absence of a type suffix.	7D, .01f

d. **Boolean literals:**

There are two Boolean literals' true or false

1—true

0—false

e. **String literals:**

It is sequence of characters between pair of double quotes.

Ex—"Hello Java" "result =0.01"

f. **Null literals:**

- It represents null value '\0'.
- It is specified as null it means the end of string.

Punctuators-- (Separators)

- It is a type of token that has syntactic and semantic meaning to the compiler.
- It is used for grouping and separating the numeric and non-numeric data.

Name	Use
()	)Parameter in method definition, contains statements for condition etc.
{}	Used to define a code for method and classes.
[]	Used for array declaration.
	Used to show the separate statement
	Used to show the separation in identifier in variable declaration
	Used to show the separate package name from sub packages and classes. Separate variable and method from reference variable.

Ex—(1) int a, b=10 ;

Here (comma) is a punctuation

(2) ;(semicolon) is used to separate statement

Operators

Operators are those tokens that perform a specific tasks/ computation when applied on variable or some objects in an expression.

They are ---

- Arithmetic operators
- Logical operators
- Relational operators
- Assignment operators
- Conditional operators
- Increment and decrement operators
- Bitwise operators

1. Arithmetic operators:

These operates are used in mathematical expressions. Those operators can operate on built-in data types of java.

Operator	Name	Description	Example
+	Addition	It adds the values of either sided operands	A + B =15
-	Subtraction	Subtracts the value of right handed operand from left hand operand's value.	A —B = —5
*	Multiplication	Multiplies the value of either sided operands.	A* B= 50
/	Division	Divides left hand operand by right hand operand.	B / A= 2
%	Modulas	Divides left hand operand by right-hand operand and returns remainder.	B % A = 0

2. Relational operator:

Relational operators compare two operands to one another. The relational operators are =, <, >, <= and >=. When used in an expression, they all return a Boolean value which states the result of comparison**. Relational operators are sometimes called comparison operators.**

Operator	Name	Description	Example
==	(Equality)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A = B) is not true
!=	(Not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A! = B) is true
>	(Greater than)	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.
<	(Less than)	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.
>=	(Greater than or equal to)	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.
<=	(Less than or equal to)	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.

3. **Logical operators:**

These operators are used to make a decision on two conditions. Logical operators are typically used with Boolean values and what they return is also a Boolean value. Logical operators are mainly used to control program flow.

Operator	Name	Description	Example
&&	Logical AND	Returns TRUE if both the conditions are true; FALSE otherwise	((A<B)&&(C<B)) returns FALSE
II	Logical OR	Returns TRUE if any of the two conditions is true	((A<B)II(C<B)) returns TRUE
!	Logical NOT	Works on single operand and reverses its state	!((A<B)&&(C<B)) returns TRUE

4. **Conditional Operator:**

Conditional operator is used when one expression and two statements are given, it is a ternary operator. The syntax of this operator is like,

Syntax: expl? exp2 : exp3;

Where exp1 is an expression and exp2 and exp2 are statements.

For example:

a==3; b=4; c=(a<b)? a:b;

c= (3<4)? 3:4; c=3

In above example, the compiler will firstly evaluate the condition (a<b), (3<4), which is true. So it will return 3.

Nested conditional operator the conditional operator can be nested.

For example, let you have three numbers a, b, c. if you want to find smallest number among these numbers, then the conditional operator will be used like.

int a, b, c, min;

min= ((a<b)? ((a<c)? a : c):((b<c)?b:c));

Let a=5, b=3, c=4

Min=((5<3)?((5<4)?5:4):((3<4)?3:4));

The compiler will evaluate the first condition (5<3), it is false. So, the control will pass to the conditional operator ((3<4)? 3:4) and it will be evaluated it is true. So it will return the value 3.

5. **Increment and decrement operator:**

The increment operator is used to increase some value by 1. The decrement operator is used to decrease some value by 1. Both are unary operators usually these operators are used in, for and while loops.

Increment operator ++

Decrement operator- -

Expression	Process	Example	Result
A++	Add 1 to variable after useint	A = 2, B; B = A++;	A = 3 B = 2
++A	Add 1 to the variable before use	int A = 2, B; B = ++A;	A = 3 B = 3
A — —	Subtract 1 from a variable after use	int A = 2, B; B =A — —;	A =1 B = 2
— — A	Subtract 1 from a variable before use	int A = 2, B; B =— — A;	A =1 B = 1

6. Bitwise operators:

A bitwise operators works on bits and perform bit-by-bit operation. These operators manipulate the values of data at bit level. These operators are generally used for testing and shifting of bits.

Operator	Name	Description	Example
&	bitwiswe AND	binary AND operator copies a bit to the result, if it exists in both operands	(A&B) gives 12, which is 0000 1100
I	bitwiswe OR	binary OR operator copies a bit, if it exists in either operands	(A I B) gives 61, which is 0011 1101
^	bitwiswe XOR	binary XOR operator copies the bit, if it is set in one operand but not both	(A^B) gives 49, which is 0011 0001
~	bitwiswe Complement	binary 1's complement operator is unary, flips the bit	(~A) gives -61, which is 1100 0011
<<	Left Shift	binary Left Shift operator. The left operand valuee gets moved left by the number of bits specified by the right operand.	A<< 2 gives 240, which is 1111 0000
>>	Right Shift	binary Right Shift operator. The left operand valuee gets moved right by the number of bits specified by the right operand.	A>> 2 gives 15, which is 1111
>>>	Right Shift	Shift Right Zero Fill operator. The left operand valuee gets moved right by the number of bits specified by the right operand and shifted values gets filled up with zeros	A>>> 2 gives 15, which is 00001111

7. Assignment operators:

These operators are also known as **shorthand operators**. These operators are used to assign the value of an expression to a variable.

Operator	Name	Description	Example
=	Assignment operator	assigns values from right side operands to left side operand	C= A + B; assign value of A+B to C
+ =	Add and Assignment operator	adds right operand to the left and assigns the result to the left operand	C + = A is equivalent to C= C +A

- =	Subtract and Assignment operator	subtracts right operand from the left and assigns the result to the left operand	C - = A is equivalent to C= C -A
* =	Multiply and Assignment operator	multiplies right operand with the left and assigns the result to the left operand	C * = A is equivalent to C= C *A
/ =	Divide and Assignment operator	divides lefft operand with the right and assigns the result to the left operand	C / = A is equivalent to C= C /A
% =	Modulus and Assignment operator	takes modulus using two operands and assigns the result to the left operand	C % = A is equivalent to C= C %A

8. Unary, binary and ternary operators in java

Unary operator: These are the operators which work on single operands.
For example: !, -, ++, _, (), operator, unary +and unary (- Unary, binary and ternary operators in java
Binary operators: These are the mostly used operators. These operators work on two operands. Binary operators include arithmetic operators (+, -,*, /, % etc.)
Ternary operators: Operator that works on three operands is known as ternary operator.
Conditional operator (? :) is the example of ternary operator

Precedence of Java operators:

When in an expression more than one operator are present then to decide which operator will be applied first and which one will be the next, we use precedence and associativity. Precedence is the order in which operators are applied some operators have higher precedence while some others have lower precedence.

Precedence order:

When two operators share an operand the operator with the higher precedence goes first. For example, 1+2*3 is treated as 1+ (2*3) whereas 1*2 +3 is treated as (1*2)+3 since multiplication has a higher precedence than addition.

Associativity:

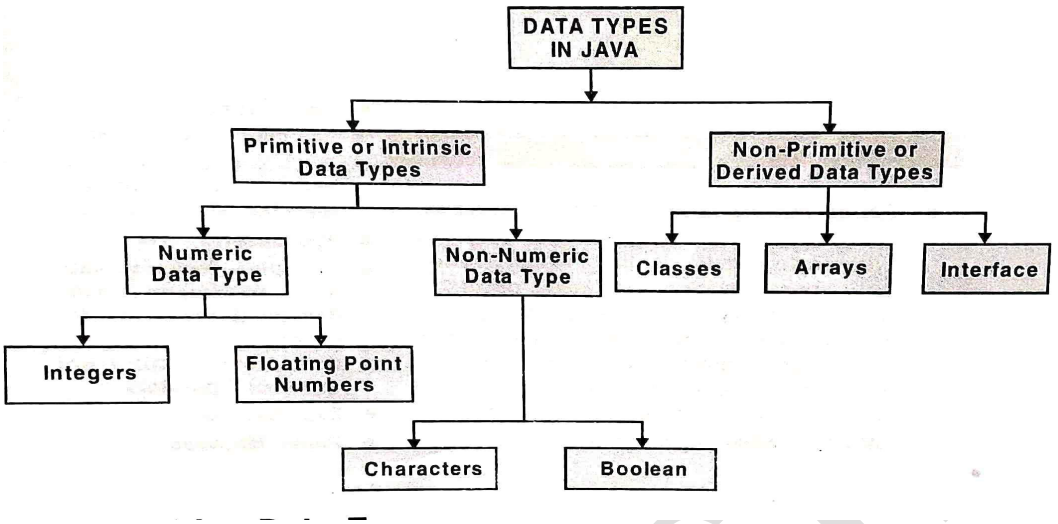
When an expression has two operators with the same precedence, the expression is evaluated according to its associatively.
For ex x=y=z=17 is treated as x=(y=(z=17)), leaving all three variables with the value 17, since the =operators has right –to-left associatively. On the other hand, 72/2/3 is treated as (72/2)/3 since the /operator has left to right associativity.

Precedence	Operator	Description	Associativity
1	() [] .	Parentheses Array subscript Access object method	Left to Right
2	++ -- + - ! ~	Postfix Prefix increment Postfix Prefix decrement Unary plus Unary minus Logical NOT Bitwise NOT	Right to left
3	(type)	Unary type cast	Right to left
4	* / %	Multiplication Division Modulus	Left to right
5	+ - +	Addition Subtraction String Concatenation	Left to right
6	<< >>	Bitwise left shift Bitwise right shift with sign extension	Left to right
	>>>	Bitwise right shift with zero extension	
7	< <= > >= instance of	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
9	&	Bitwise AND	Left to right
10	^	Bitwise exclusive OR	Left to right
11	|	Bitwise inclusive OR	Left to right
12	&&	Logical AND	Left to right
13	||	Logical OR	Left to right
14	? :	Ternary conditional	Right to left
15	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

DATA TYPES:

Data type describes the type of data that a variable can store along with the set of operations performed on it.

- The programming language java provides all types of data.



Primitive data types:-

- Primitive data types are built in data types.
- If supports 8 types of primate data type. They fall under 4 categories.
  - Numeric
  - Floating point type
  - Character type
  - Boolean type

1. Numeric data type:

- These store only integer values.

They are of four parts—

I. **Byte:**
   - It is smallest numeric data type.
   - It uses 8 bits to represent a number
   - It has range value -128 to 127
   - It is of **1 byte**

II. **Short:**
   - It has small range of values ranging from -32768 to 32767
   - It uses **2 bytes** to represent a number.

III. **Int:**
   - It is commonly used 32-bit signed integer range from -2147483648 to +2147483647
   - It is of **4 bytes**.

IV. **Long:**
   - It is a 64-bit signed integer having a wide range of values.
   - It is of **8 bytes**.

2. Fractional / Floating data type(numeric)

- Data types which stores fractional numbers are known as floating data type.
- The other name of these data type is known as floating point data types.
- They are of two types of
  - Float
  - Double

Float:

- It holds floating point numbers of 32 bit.
- It has +ve or-ve sign. It has greater range of integers ranging from +ve to –ve.
- It is of **4 bytes** ranging from -3.4×10^{38} to 3.4×10^{38}
- It is of **8 digits precision** (no of digits allowed after decimal).

Double:

- It is extended version of float data type.
- It couples doubles memory with larger range i.e **8 bytes.**
- It is of **15 digits precision** and ranges from -1.7×10^{308} to 1.7×10^{308}
- **Example of double**-complex math function such as sin (), sqrt() etc.

3. **Character data type (Non-numeric)**
   - The data type is of **2 types** which store in code characters**.**
   - These characters are stored in bring from as integers.
   - These characters are assigned a particular value according to ASCII code.
   - Only java supports Unicode characters. It represents all languages of world.
   - It is of 16- bits and venues from 0 to 65535.
   - It uses char keyword.
4. **Boolean data type (Non-numeric)**
   - It is used to represent logical value.
   - It tests a particular condition i.e. true or false.
   - It reserves 8-bit i.e **1 byte** storage space but it uses only 1-bit.

Referenced data types (Non-primitive data types)

- They are made from Non-Primitive data types.
- They are defined by the programmer according to the requirement of programmers.
- Ex –classes, arrays, interface.
- It stores memory address of an object.

Arrays:

- It is a collection of fixed size finite numbers of objects and values that belong to same data type.
- It is a collection of data storage locations.

Class:

- Class is collection of objects. It is user defined data type.
- A class may be defined as group of objects with some operations and attributes.

Interface:

- It is group of methods to provide basic functionality to classes to share common behavior.

Difference between Primitive and Non primitive data type

Primitive:

- These are built-in data types.
- There are 8 primitive data types provided by java.
- These types are handled value.

Non-primitive data type:

- They are user defined data types.
- The non-primitive data types in java are objects and arrays.
- These types are handled by reference.

String in java:

- They are the sequence of characters.
- String can be declared and created as follows:

  String name;

  (or)

  String name= new String (“string”);

  For Ex-a string “information technology” is declared below

  String STR;

  STR =new string (“information technology”);

Variable:

- It is the name of location where the information is stored.
- It is an identifier which holds data or another one.
- It is an identifier whose value can be changed at the exaction of program.

Declaring a variable:

- All variables must be declared before they are used in java program.
- It is declared by providing the data type of a variable.

Syntax: **data_type variable_name**;

For ex:- To declare character variable:- char ch;

To declare integer variable:- int x**;** To declare floating point variable: float a;

Variable Naming:

It defines a set of rules and regulations to decide the name of variable.

Rules to name a variable:

- Variable name are case-sensitive.
- No spacing is allowed with in variable names.
- No special symbol can be used in variable names such as!, 0, #, & etc.
- A variable name cannot start with numeric value or underscore (_) symbol.
- A variable name must not be a keyword or reserved word.
- Every variable name should start with an alphabet.

Assigning values to variables:

- Value is assigned to a variable after it is declared.
- Assigning a value to a variable is known as initialization of variable.
- To initialization a variable we use an assignment operator.

General from of initialization (declaring variable)

<datatype> <variable name> = constant value;

float salary=54000;

char salute='Namaste';

int n=54;

Note: char values are whether single quotation marks but not the numeric values.

Constant in java:

- Those values which never get changed.
- Day have 24 hours, the value of pi will always 3.141.
- While programming these values remain in same way these variables are known as constant.
- Constant are always declared by final keyword because final is a reserved keyword which the compiler that the value will remain unchanged.

For ex:- int hours=24

So it can be used as:-

final int hours=24

Note: It is declared only once in the program.

Structure of program

Package detail ——→ import java.io.*

Class classname ——→ class my class

{

Data members ——→ int a, b, c;

User defined methods ——→ void display ()

Public static void main (string args[])

{

Block of statements; ——→ System.out.println ("hello java");

}

}

- A package is a collection of classes, interfaces and sub packages.
- A sub package contains collection of classes, interfaces and sub-sub packages.
- Class is a key word used for developing user defined data type.
- "class name " represent a java valid variable name of the class
- User defined methods represents which are meant for performing the operations either once or each and every time.
- Each and every java program starts execution from main () method and main method () is known as program driver.
- Main () method of java doesn't return any value and hence its return type is void.
- Main () method is must be all used by every java programmer so that access specifier must be public.

- Block of statements represents set of executable statement which are called user-defined methods.
- The file naming conversion in java programming is that whichever class is containing main() method that class name must be given as a file name with an extension.java

Main () method

- Main () is string execution block of a java program.
- Java program start their execution from main method.
- If any class contain main () method of is known as main class.

Syntax-

```
public static void main (String arg[])
{
----------;
----------;
}
```

public:

It is a keyword if it is produced by main () method, the scope is available anywhere it means main () method can be executed from anywhere.

static:

It is a keyword if tit is preceded by any class properties for that memory is allocated only once in the program.

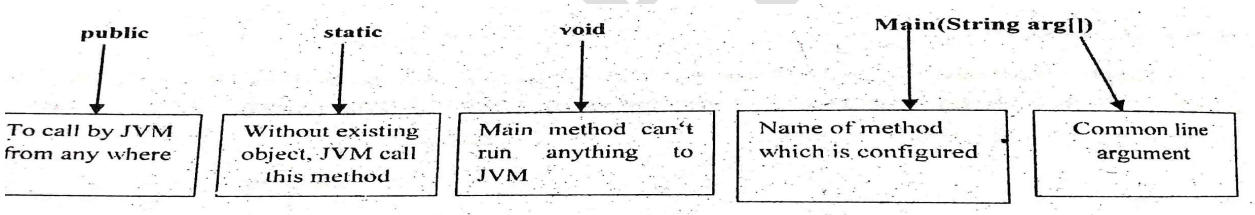
void:

It has no return type, if preceded by main () method it will never return any value to the operating system.

String args[]:

It is the string the array used to hold command line arguments in the form of string values.

Diagram



Programs (basic):

1. **W.A.P. in java to print your name**

```
// java hello world example
public class my name
{
public static void main (String args[])
{
        System.out.println("my name is xyz");
}
}
```

2.W.A.P in java to find sum of 2 nos.

```
public class add
{
public static void main (String args[])
{
    int a=10, b=20,c;
    c=a+b;
    System.out.println("sum="+c);
}
}
```

Parse methods:

- Sometimes there is a need to use numeric values in text type components.

- Suppose we want to read it of a person in a text field but text field returns the text output so we need to convert the text data into numeric data.
- For this purpose parse () methods are used

1. Integer object method:

Integer is a wrapper class that holds int variable.

parseInt ()/Integer.parseInt() method:

Returns an int value specified by the string parameter.

Syntax:

Integer.parseInt(Strings);

Ex:

```
public class example
{
Public static void main (String args [])
{
int x =Integer. parseInt ("123");
System.out.println(x);
}
}
```

Output:

123

2. parseDouble() method:

- Double is a wrapper class that holds a double variable.
- Parse double () method is used to return a double type value for the value represent by a specified string.

  Double i = Double.parseDouble (String args())

```
class doubleparse
{
public static void main (String args[ ])
double x= Double.parseDouble(555F);
System.out.println(x);
}
}
```

Output

555F

3. parseFloat()method:

- It is used to convert string to float point type value.

Syntax:

float x=Float.parseFloat (Strings);

How to develop a program in GUI method

Step-1: click file→new file.From the categories panel select Swing GUI forms and from the file types panel, select jframe forms→click next.Give the class name→click finish button

Step-2 : Drag the required jLabels control, jTextfields control and jButtons from the Palette window into design window.

Step-3 : right click on the jTextfields and change the variable name.

Step-4 : now double click the button adds and writes the following code.

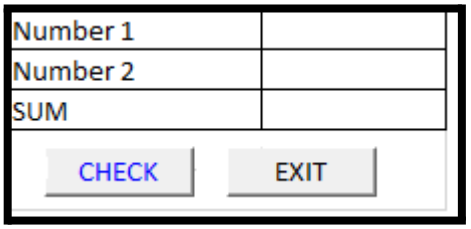
Step-5:then double click on the button exit and the following line is written

System.exit(0);

Step-6:click Shift +F6 to run the application/program code.

1. **Develop a GUI form to perform simple addiction problem display output on text field control.**

   Design :



Setting Property

Control Name	Property Name	Property Value
jFrame	Text	addition problem
jLabel1	Text	Number1
jLabel 2	Text	Number 2
jLabel 3	Text	SUM
jTextField 1	Text	empty
	Change variable name	n1
jTextField 2	Text	empty
	Change variable name	n2
jTextField 3	Text	empty
	Change variable name	n3
jButton 1	Text	ADD
jButton 2	Text	EXIT

Coding

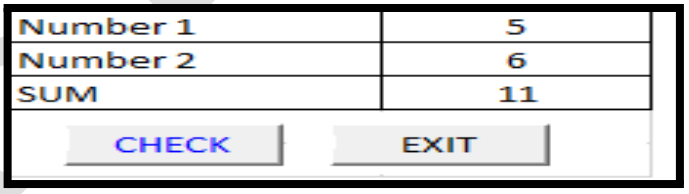
Double click the ADD button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int num =Integer.parseInt(n1.getText());
int num 2= Integer.parseInt(n2.getText());
int sum=num1 +num2;
n3.setText (“  ”+sum);}
```

Double click the EXIT button and write the following code.

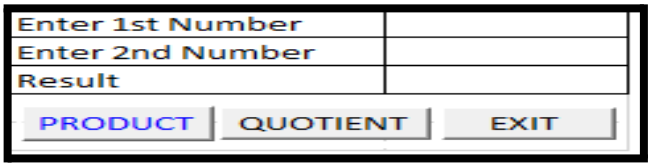
```
private void JButton2Actionperformed (java.awt.event.ActionEvent evt)
{
System.exit(0);
}
```

Output



Ex-2

Develop a GUI form on java to perform product and quotient of 2no.s.



Setting Property:

Control	Property Name	Property Value
jFrame	Text	Product and Quotient
jLabel1	Text	Number1
jLabel 2	Text	Number 2
jLabel 3	Text	Result
jTextField 1	Text	empty
	Change variable name	n1
jTextField 2	Text	empty

	Change variable name	n2
jTextField 3	Text	empty
	Change variable name	n3
jButton 1	Text	Product
jButton 2	Text	quotient
jButton 3	Text	Exit

Code:

Double click the Product button and write the following code:

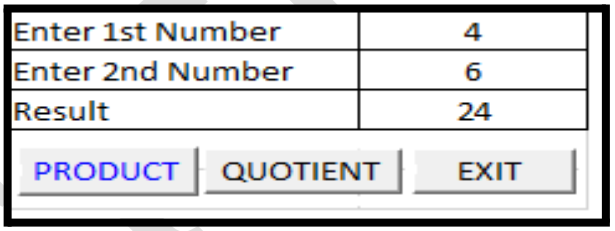
```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int a= Integer.parseInt(n1.getText ());
int b= Integer.parseInt(n2.getText ());
int product=a*b;
n3.setText (“ ”+product);
}
```

Double click the Quotient button and write the following code:

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
{
int a= Integer.parseInt (n1.getText ());
int b= Integer.parseInt (n2.getText ());
int quotient= a/b;
n3.setText (“ ”+quotient);
}
```

Double click the end button and write the following code.

```
private void jButton3ActionPerformed (java.awt.event.ActionEvent evt)
{
System.exit(0);
}
```



Java control structure with examples.

Control flow statements:

- It determines an order in which statements used in a program are executed.
- Its night because a statement to be executed once, many times or not at all.
- The following control statements are:
  - Decision structure
  - Looping structure
- Java programming language provides following types of decision-making statements.
  If structure
  If…….. else structure
  Switch structure

if structure:

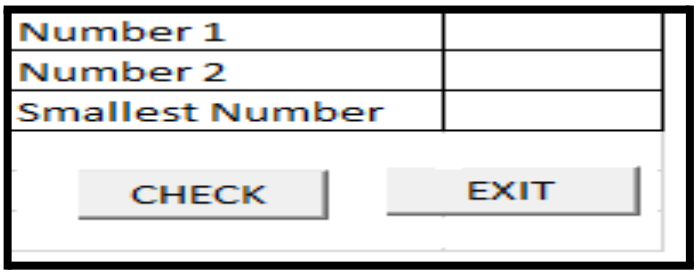
- It is the basic form of selection statements which executes a single block of statements if the condition is true.

Syntax:

```
if (expression)
{
Statements;
}
```

Examples demonstrating if statement

Write a program for if statement showing how to find the smallest number between two.



Control	property name	property value
jFrame	Text	smallest number
jLabel1	Text	Number1
jLabel 2	Text	Number 2
jLabel 3	Text	Smallest number
jTextField 1	Text	empty
	Change variable name	n1
jTextField 2	Text	empty
	Change variable name	n2
jButton 1	Text	CHECK
jButton 2	Text	EXIT

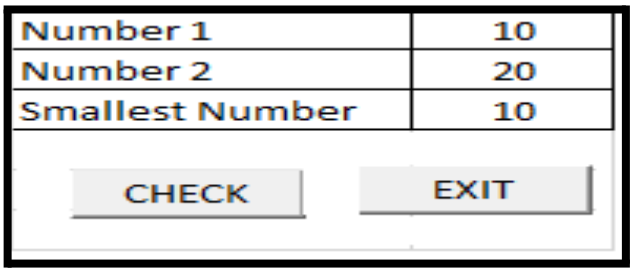
Step-1 start new file and name it smallest number.
Step-2 click and drag three j label control, three j text field and two j button controls in the form and set the property editor in the right-hand side.
Step-3 change the variable name of the j text field by right click on the respective j text field from inspector window on the lower left pane of net bean window.
Step-4 double click on find button which will generate a procedure called button click action performed and write the following lines as given below.

```
int a= Integer.parseInt(n1.get text ());
int b=Integer.parseInt(n2.get text ());
int smallest =0;
if (a<b)
{
smallest=a;
n3.setText (“    ”+smallest);
}
```

Step-5 double clicks on the exit button and writes the following statement given below

```
System.exit(0);
```

Step-6 press ctrl+s to save the file.
Step-7 press shift+f6 to run the file



Ex:

Write a code in GUI to find smallest of public class smallest two numbers.

Double click the check button and write the following code:

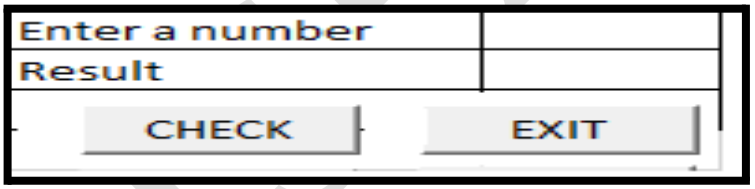
```
private void jButtonActionPerformed (java.awt.event.ActionEvent evt)
{
int a= Integer.parseInt(n1.getText ());
int b=Integer.parseInt(n2.getText ());
int smallest =0;
if (a<b)
{
Smallest=a;
n3.setText (“    ”+smallest);
}
```

Double click the end button and write the following code.

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt){
System.exit(0);
}
```

Ex:

Write a program to check whether the number is even or not.

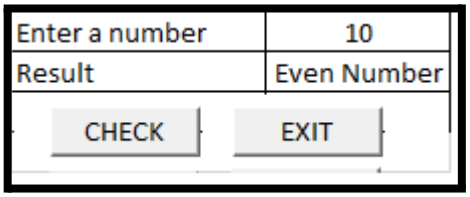


Double click the check button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int a=Integer.parseInt (n1. getText ());
int even=0;
if (a%2= =0)
{
even=a:
n2.setText (“      ”+even);
}
```

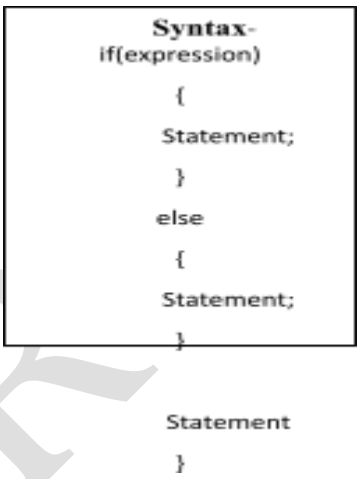
Double click the end button and write the following code.

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
{
System.exit(0);
}
}
```



If else statement

- It is the type of selection statement in which single statement or a group of statements enclosed within the if condition is executed when it is true.
- Otherwise the else block is executed.



Switch-case statement:

- It is an alternative used for if-else condition.
- If a programmer has make number of decision and all the decisions depend on the value of variable, then switch care is used instead of if-else condition.

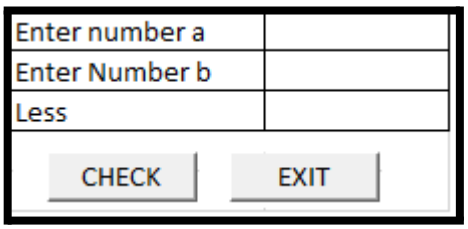
Syntax:

```
Switch (expression)
{
case1:
Action 1 statement;
break;
case2:
Attion2 statement;
break;
--
--
--
--
--
case n
Action n statement;
break;
default:
Default statement;
}
```

Note- variable used in a switch statements can only be integers line byte, short, int, char,

If-else example:

Write a code for GUI application to find number of students between two different classes of college



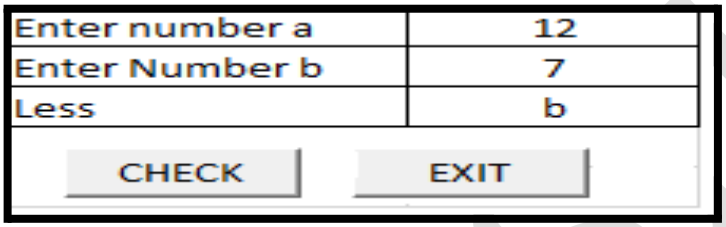
Double click the check button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
    {
    int a= Integer.parseInt (n1.getText ());
    int b=Integer.parseInt (n2.getText ());
    int less=0;
    if (a<b)
```

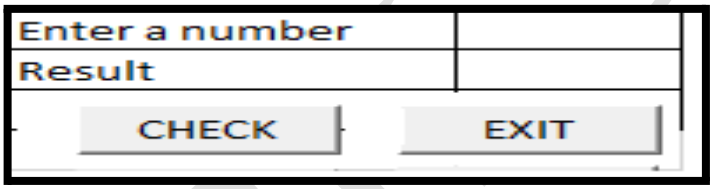
```
{
less=a;
n3.setText (“    ”+less);
}
else
{
less=b;
n3.setText (“   ”+less);
}
}
```

Double click the end button and write the following code.

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
{
System.exit(0);
}
```



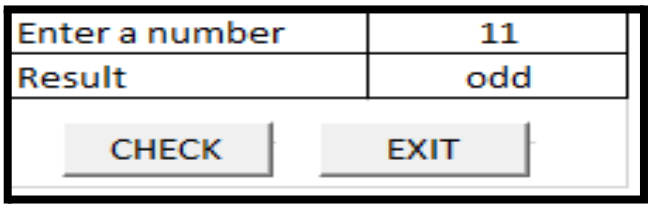
Write a GUI program to find whether the number is even or odd.



Double click the check button and write the following code:

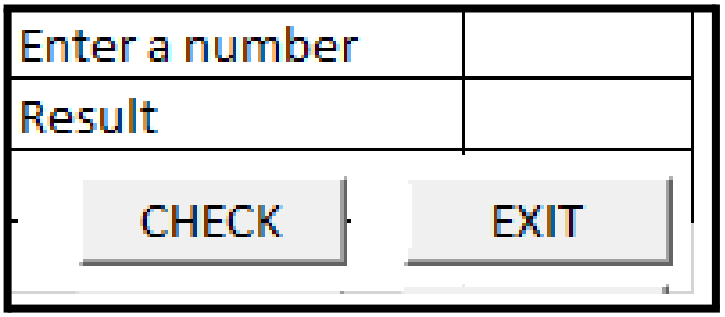
```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int a= Integer.parseInt (n1.getText ());
if (a%2= =0)
{
n2.setText (“the no. is even”);
}

else
{
n2.setText (“the no. is odd”);
}
}
```



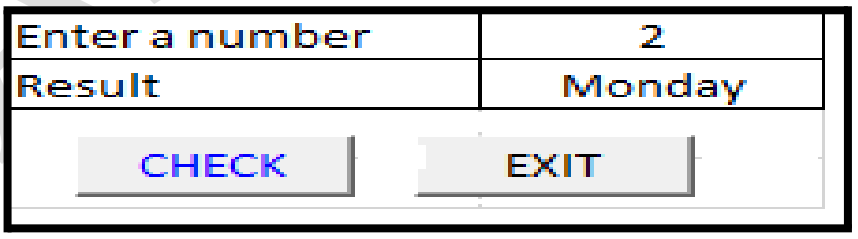
<u>Switch structure example:</u>

Write a code for GUI application using net beans ide to display the name of the day corresponding to an input number.




```
int d= Integer.parseInt (n1.getText ());
switch (d)
{
case 1:
        n2.setText ("Sunday");
        break;
case 2:
        n2.setText ("Monday");
        break;
case 3:
        n3.setText ("Tuesday");
        break;
case 4:
        n4.setText ("Wednesday");
        break;
case 5:
        n5.setText ("Thursday");
        break;
case 6:
        n2.setText ("Friday");
        break;
case 7:
        n2.setText ("Saturday");
        break;
default:
        n2.setText("wrong chore");
}
}
```

Output



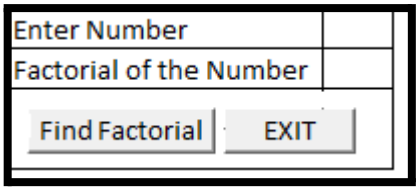
Loop construct (looping structure):

- We require a set of statements to be executed number of times by changing the values of one or more variables each time to obtain a different result.
- Java provides the following loop constructs for
  1. While loop
  2. Do while loop
  3. For loop

While loop:

- A while loop is used for repetitive execution of a program.
- The statements may be a single statement or block of statements.

Than 20 write a program using non-GUI method to print natural numbers less
Double click the check button and write the following code:



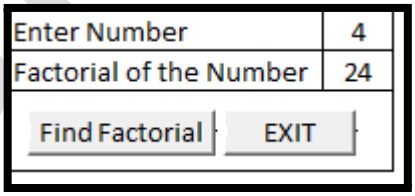

```
 private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int x=10;
while (x<10)
{
System.out.println("value of x:"+x);
x++;
}
}
```

Double click the end button and write the following code.

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
{
System.exit(0);
}
```

Demonstration of while statement to find the factorial number.

```
int n = Integer.parseInt(n1.getText());
inti;
int fact=1;
while (i<=N)
{
fact=fact*i;
i++;
}
n2 .setText ("    "+fact);
```



do……while
Syntax:

```
.do
{
// statements
} while (Boolean expression);
```

- The impression appears at the end of the loop so the statements in the loop execute once before the Boolean is tasted.
- If impression is true, the content jumps back up to do statement and the statements in the loop execute again.

Program:
Double click the check button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
int x=10;
do
```

```
{
System.out.println (value of x: +x);
}
while(x<10);
}
```

for loop:

- A for loop is a repetition control that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for(initialization; expression; increment/decrement)
{
    // statements
}
```

Example:

Double click the check button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
    {
    for (int x=100; x<110; x=x++)
    {
    System.out.println ("value of x:" +x)
    }
    }
```

Output:

```
Value of x: 100
Value of x: 101
Value of x: 102
                ...
                ...
Value of x: 109
```

for loop:

Double click the check button and write the following code:

```
private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
    {
    inti;
    for (i=1;i<=5;i++)
    {
    System.out.println("  "+i);
    }
    }
```

Output:

1
2
3
4
5

PROGRAMMING GUIDELINES

Running And Debugging Programs:

Running a program is also known as executing process for finding whether the current coded program runs correctly or not and produces the output we want.

Debugging is a process of finding and reducing the number of errors and defects, in a computer program.

A debugging process can be divided into three main parts.

1. Identifying a bug
2. Classifying a bug
3. Fixing the bug

Error: An error is a flow, fault or failure in a computer program that causes it to produces an incorrect or unexpected result. Or to behave in unintended ways.

Broadly, there are three types of errors:

1. **Compile-time Error**

All the errors that are detected and displayed by the Java compiler are known as compile-time errors. Whenever the compiler displays an error, it will not be able to run.

There are two categories of compile-time errors: Syntax errors & Semantic errors.

(i)Syntax error

When a formal set of rules defined for writing a program in a particular language is not followed then error raised is known as syntax error. Syntax errors occur when syntax rules of any programming language are violated. Net-Beans highlighted syntax errors in design state itself using the error indicator.

Example of syntax errors are missing semicolon, parenthesis etc.

(ii) Semantic error

This type of compile time errors indicates an improper use of Java statement. These errors occur when statements are not meaningful. The word "semantics" relates to the meaning of words, sentences, or programs.

For examples, use of an undeclared variable comes under semantic errors.

2. Run-time Error:

Run-time errors occur during the execution of a program. These errors will result the abnormal termination of program. Examples of run-time errors are number division by zero etc.

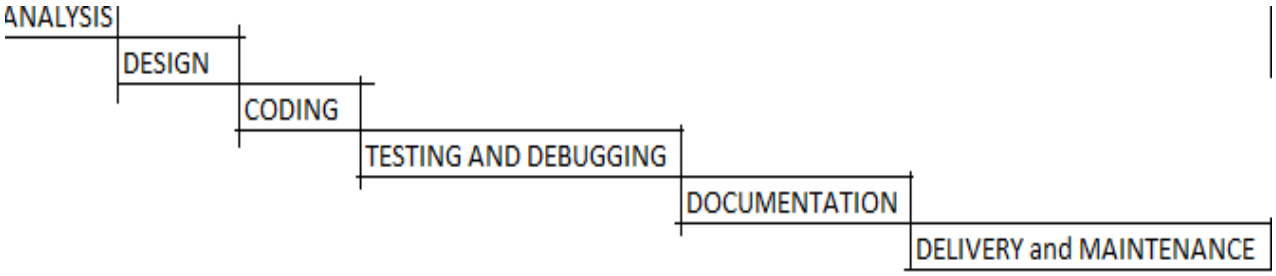
3. Logical Error:

These errors occur due to mistakes of the programmer. This is the most difficult task to find and debug a logical error.

For example, when a wrong formula is used to calculate any mathematical expression gives logical error.

Stages of a simple GUI Application Development

Various steps involved in the devopment of a new application are as follows.



1. Analysis: This phase involves- the following steps:

(a) in-dept understanding of the problem

(b) deciding the requirements of the problem

(c) Jotting down possible inputs and outputs that are required for obtaining the desired solution.

2. Design: This phase involves planning of step-by-step procedures required to slove a given problem. At this stage a detailed design of the following components is to be completed:

(a) Inputs (b) Output (c) User Interface (Forms) (d) Modular Components

(e) Algorithms

3. Coding: This phase involves actual writing of programs using appropriate programming languages. A good programmer will make an optimum code, which is readable, easy to understand and maintain with appropriate error handing, comments and indentation.

4. Testing and Debugging: Testing means the process of executing the application with possible set of input data in order to find probable errors. Debugging means correction of those errors in the application.

5. Documentation: Documentation means the instructions and information about the usage of the application. Providing the documentation makes it easier for the end user to understand the functionally of the application.

6. Application Delivery and Maintenance: The completed software is packaged with full documentation and delivered to the end users. The maintenance involves the rectification of previously undetected errors and changes that are to be made for the enhancement of the functionality of the application.