Fundamentals of Java Class 11 Notes

Fundamentals of Java Class 11 Notes

Programming in Java is simple to write, compile, and debug. Reusable code and modular programmes can be made with its assistance. As few implementation dependencies as possible are built into Java, an object-oriented, class-based programming language.

NetBeans IDE

For the development of applications on the Windows, Mac, Linux, and Solaris operating systems Net Beans platform is best, NetBeans is GUI based free and open source software. NetBeans helps to developing Java based software easily, It also support many other language like HTM5 etc.

Components of NetBeans IDE

The user can easily interacts with NetBeans COMPONENTS which helps the user to design the software, NetBeans components include jlabels, jbuttons, jtextfields, etc. There are two types of controls –

Parent or container controls – Controls that serve as a parent or container serve as the backdrop for other controls. For example – Frame.

Child controls – Child controls are controls that are contained within a container control. Text field, label, button, etc. are a few examples.

Note – We use the drag and drop feature of NetBeans to place components on the form to design an effective interface for our applications.

Creating a New Project

1. Select New Project from the File menu. You can also click the New Project button in the IDE toolbar.
2. In the Categories pane, select the General node. In the Projects pane, choose the Java Application type. Click the Next button.
3. Enter the name of the project in the Project Name field and specify the project location. Do not create a Main class here.
4. Click the Finish button.

Properties in NetBeans IDE Control

jButton Properties & Method

Properties –

1. Background – Sets the background color.
2. Enabled – Contains enabled state of component – true if enabled else false.
3. Font – Sets the font.
4. Foreground – Sets the foreground color.
5. horizontal alignment – Sets the horizontal alignment of text displayed on the button.
6. Label – Sets the display text.
7. Text – Sets the display text

Method –

1. getText() – Retrieves the text typed in jButton.
   String result=<button-name>.getText( );

2. setEnabled – Enables or disables the button.
   <button-name>.setEnabled(boolean b);
3. setText() – Changes the display text at runtime.
   <button-name>.setText(String text);
4. setVisible – Makes the component visible or invisible – true to make the component visible; false to make it invisible. <button-name>.setVisible(boolean aFlag);

jTextField Properties and Method

Properties –

1. Background- Sets the background color.
2. Border – Sets the type of border that will surround the text field.
3. editable – If set true user can edit textfield. Default is true.
4. enabled – Contains enabled state of component- True if enabled else false.
5. font – Sets the font.
6. foreground – Sets the foreground color.
7. horizontalAlignment – Sets the horizontal alignment of text displayed in the textField.
8. text – Sets the display text
9. toolTipText – Sets the text that will appear when cursor moves over the component.

Method –

1. getText() – Retrieves the text in typed in jTextField.
   String result=<textfield-name>.getText( );
2. isEditable() – Returns true if the component is editable else returns false.
   boolean b=<textfield-name>.isEditable( );
3. isEnabled() – Returns true if the component is enabled,else returns false.
   boolean b =<textfield-name>.isEnabled( );
4. setEditable – Sets whether the user can edit the text in the textField. true if editable else false.
   <textfield-name>.setEditable(boolean b);
5. setText() – Changes the display text at runtime.
   <textfield-name>.setText(String t);
6. setVisible() – Makes the component visible or invisible – true to make the component visible; false to make it invisible.

<textfield-name>.setVisible(boolean b);

jLabel Properties and Method

Properties –

1. background – Sets the background color.
2. enabled – Contains enabled state of component- true if enabled else false.
3. font – Sets the font.
4. foreground – Sets the foreground color.
5. horizontalAlignment – Sets the horizontal alignment of text displayed in the component.
6. text – Sets the display text

Method –

1. getText() – Retrieves the text in typed in jLabel.
   String result=<label-name>.getText();
2. isEnabled() – Returns true if the component is enabled,else returns false.
   boolean b=<label-name>.isEnabled();
3. setText() – Changes the display text at runtime.
   <label-name>.setText(String t);
4. setVisible() – Makes the component visible or invisible – true to make the component visible; false to make it invisible. <label-name>.setVisible(boolean b);

jTextArea Properties and Method

Properties –

1. background – Sets the background color.
2. columns – Sets number of columns preferred for display.
3. editable – If set true user can edit textfield. Default is true.
4. enabled – Contains enabled state of component- true if enabled else false.
5. font – Sets the font.
6. foreground – Sets the foreground color.
7. lineWrap – Indicates whether line of text should wrap in case it exceeds allocated width.(Default is false)
8. rows Sets – number of rows preferred for display.

9. text – Sets the display text
10. wrapStyleWord – Sends word to next line in case lineWrap is true and it results in breaking of a word, when lines are wrapped.

Method –

1. append() – Adds data at the end.
   <textarea-name>.append(String str);
2. getText() – Retrieves the text in typed in jTextArea.
   String str = <textarea-name>.getText();
3. isEditable() – Returns true if the component is editable else returns false.
   boolean b = <textarea-name>.isEditable();
4. isEnabled()- Returns true if the component is enabled, else returns false.
   boolean b = <textarea-name>.isEnabled();
5. setText() – Changes the display text at runtime.
   <textarea-name>.setText(String t);

jPassword Properties

1. background – Sets the background color.
2. font – Sets the font.
3. foreground – Sets the foreground color.
4. text – Sets the display text
5. echoChar – Sets the character that will be displayed instead of text.

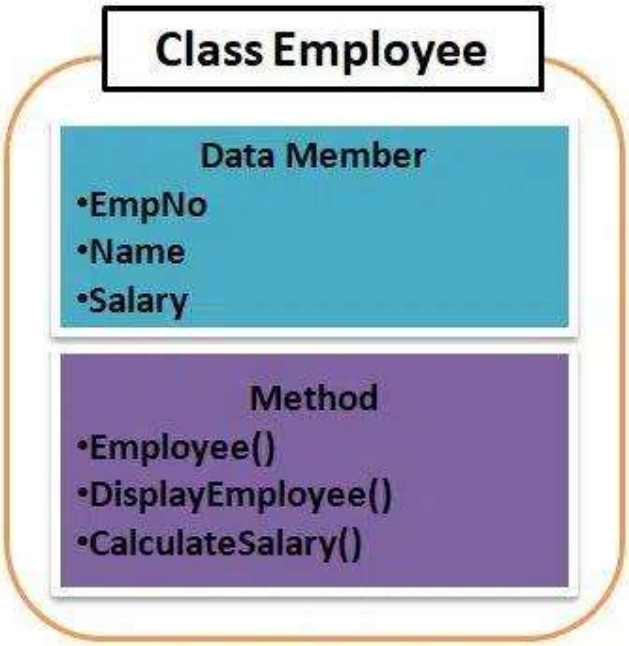
Object Oriented Programming

Object Oriented Programming follows bottom up approach in program design and emphasizes on
safety and security of data. It helps in wrapping up of data and methods together in a single unit
which is known as data encapsulation. It arranges the architecture of software around data or objects rather than functions and logic. An object is a data field with particular characteristics and behaviour.

The major components of Object Oriented Programming are as follows:
1. Class
2. Object
3. Data Members & Methods
4. Access Specifier and Visibility Modes

Data Members and Methods

Methods and data members are found in classes. Data members declare inside the class, it is variable which we are declaring inside the class. In general, data members are kept secret so that values can only be altered at the discretion of the class function members. Data members may be private or public. A method is a function declared inside a class.



The JTextField, JLabel, JTextArea, JButton, JCheckBox and JRadioButton are all classes and the jTextField1, jLabel1, jTextArea1, jButton1, jCheckBox1 and jRadioButton1 components are all objects. The setText(), setEnabled(), pow(), substring() are all methods of different classes.

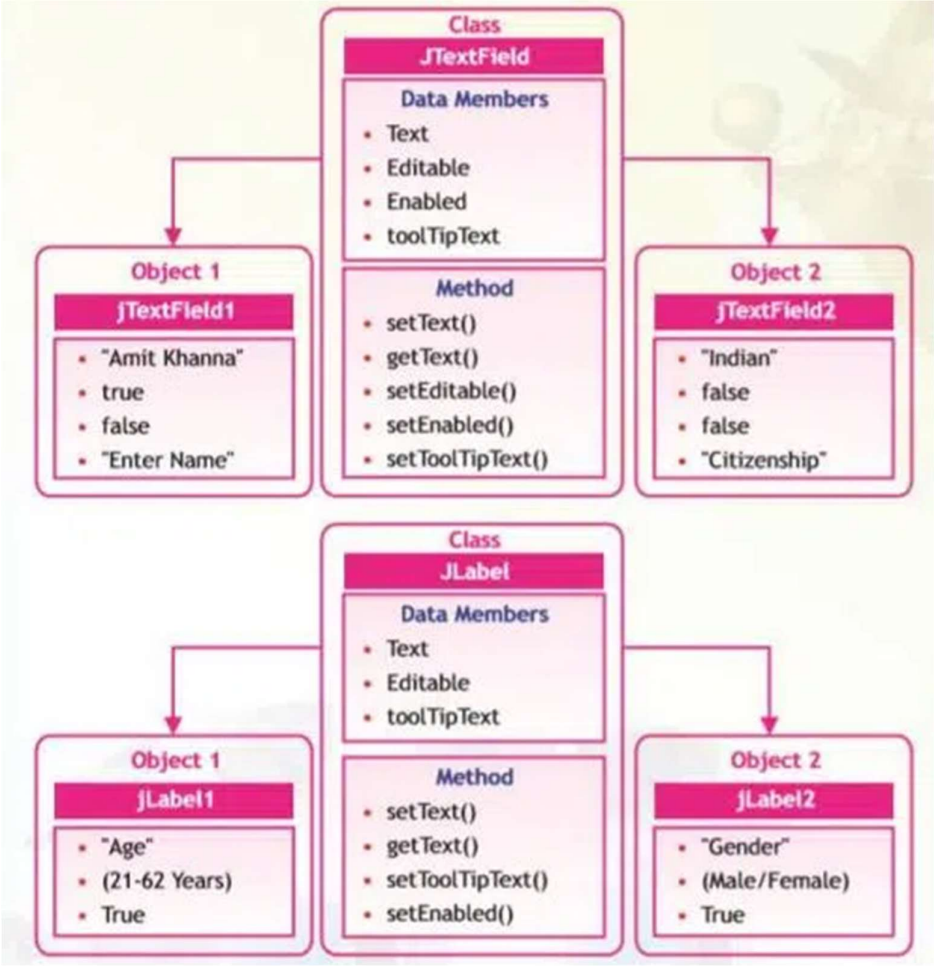


Image Source cbseacademic.nic.in

Variables

Variables are containers used to store the values for some input, intermediate result or the final result of an operation. The characteristics of a variable are:

• It has a name.
• It is capable of storing values.
• It provides temporary storage.
• It is capable of changing its value during program execution.

Data Types

Data type states the way the values of that type are stored, the operations that can be done on that type, and the range for that type.

Numeric Data Types –

These data types are used to store integer values only i.e. whole numbers only.
The storage size and range is listed below:

Name	Size	Range	Example
byte	1 byte(8 bits)	-128 to 127(-2^7 to +(2^7-1))	byte rollno;
short	2 bytes(16 bits)	-32768 to 32767(-2^{15} to +(2^{15}-1))	short rate;
int	4 bytes(32 bits)	-2^{31} to +(2^{31}-1)	int num1;
long	8 bytes (64 bits)	-2^{63} to +(2^{63}-1)	long amount;

Floating Data Types –

These data types are used to store numbers having decimal points i.e. they can store numbers having fractional values.

Name	Description	Size	Range	Example
float	Single precision floating point	4 bytes (32 bits)	($3.4x10^{-38}$) to +($3.4x10^{38}$)	float average;
double	Double precisio floating point	8 bytes (64 bits)	($1.8x10^{-308}$) to +($1.8x10^{308}$)	double principal;

Character Data Types –

These data types are used to store characters. Character data types can store any type of values – numbers, characters and special characters. The char data type value is always enclosed inside ’ (single quotes), whereas a string data type value is enclosed in “” (double quotes).

Operators

Operators are symbols that manipulate, combine or compare variables. The operators available in java are summarized below –

Assignment Operator

One of the most common operators is the assignment operator “=” which is used to assign a value to a variable. The value on the right hand side can be a number or an arithmetic expression.
For example:

```
int sum = 0;
int prime = 4*5;
```

Arithmetic Operators

These operators execute multiplication, division, addition, and subtraction. These symbols resemble those used in mathematics. Only “percent,” which divides one operand by another and returns the remainder as its result, differs from the others.

+ additive operator
– subtraction operator *
multiplication operator /
division operator
% remainder operator

Relational Operator

A relational operator is used to test for some kind of relation between two entities.

Operator	Meaning	Usage
==	equal to	Tests whether two values are equal.
!=	not equal to	Tests whether two values are unequal.
>	greater than	Tests if the value of the left expression is greater than that of the right.
<	less than	Tests if the value of the left expression is less than that of the right.
>=	greater than or equal to	Tests if the value of the left expression is greater than or equal to that of the right.
<=	less than or equal to	Tests if the value of the left expression is less than or equal to that of the right.

Logical Operator

A logical operator denotes a logical operation. Logical operators and relational operators are used together to form a complex condition. Logical operators are –

Operator	Use	Meaning
&&	a>10 && b<8	a and b are both true
||	a>10 || b<8	Either a or b is true
!	! a	a is false

Bitwise Operator

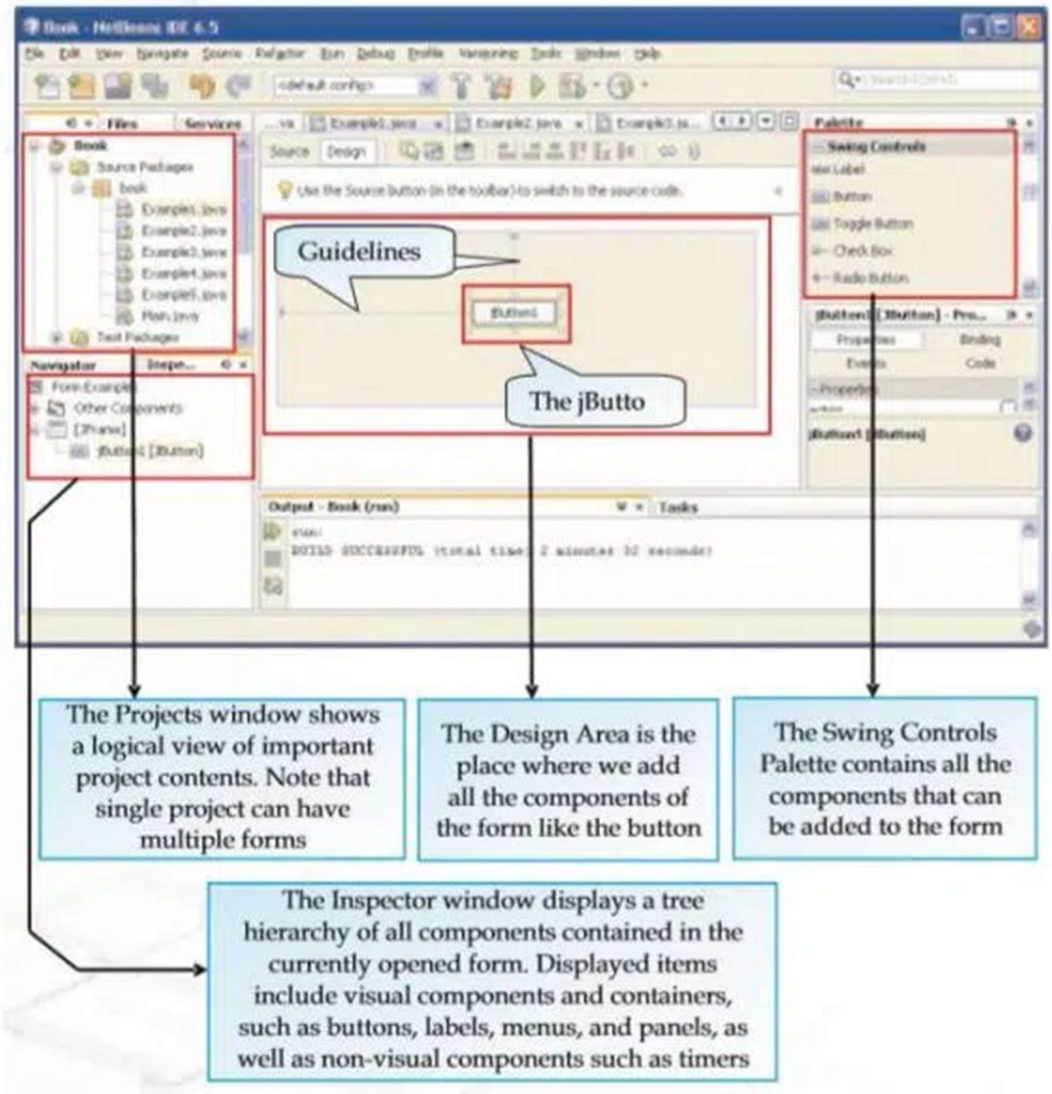
Bitwise operators can be used to change a number's individual bits. They are compatible with all integral kinds (char, short, int, etc). They are utilised while updating and searching a binary indexed tree.

Creating New Project in NetBeans –

1. Select New Project from the File menu. You can also click the New Project button in the IDE toolbar.
2. In the Categories pane, select the General node. In the Projects pane, choose the Java Application type. Click the Next button.
3. Enter the name of the project in the Project Name field and specify the project location. Do not create a Main class here.
4. Click the Finish button.

Adding a Button Component to a Form

We want to add a button so follow the given steps to add a JButton to the form –
Step 1 : In the Palette window, Select the JButton component from the Swing Controls category.
Step 2 : Move the cursor over the Form. When the guidelines appear, indicating that the JButton is positioned in desired location, click to place the button.



Attaching Code to a Form Component

The next step after placing the button is to write the code that will cause the application to close when the button is pressed. To accomplish the same thing, double-click the button to associate a code with the action, i.e. the button click. The source window is opened by double clicking on the component, and the pointer is moved to the area where code has to be added. Keep in mind that certain code is pre-generated and cannot be altered.

Executing a File

Now that the code for the first application is ready let us test our first application. To execute the application simply select Run>Run File or press Shift+F6

Changing Properties of Components

Each component of our application including the form has certain attributes associated with it. The Properties Window displays the names and values of the attributes (properties) of the currently selected component. We can edit the values of most properties in the Properties window.

We want to change the text displayed on the button. There are four ways of doing the same in the design view:
● Select the button component by clicking on it. In the Properties window highlight the text property and type STOP in the textbox adjacent.
● Alternatively select the object. Left click on the button to highlight the display text. Type STOP and press Enter.
● Select the object > Press F2 – to make the display text editable. Type in the new text and press Enter.

Displaying a Message in a Dialog Box

Now that we are familiar with the creation process, let's attempt some more experiments and see if we can get a message to appear when the button is clicked. To construct a new form with a straightforward button as shown in Figure 5.25, follow the same procedure. Change the button's attributes as needed and Make "Wish Me" the text property.

Control Structures

If Statement

The if statement enables choosing (decision-making) based on how a condition turns out. The statement that comes after the if statement will be executed if the condition evaluates to true, and the statements that come after the else clause will be executed if the condition evaluates to false. The decision or conditional statements are other names for the selection statements.

The syntax of if statement is as shown below:

Syntax:

```
if (conditional expression)
{
Statement Block;
}
else
```

```
{
Statement Block;
}
```

Points to remember about if statement –

• The conditional expression is always enclosed in parenthesis.
• The conditional expression may be a simple expression or a compound expression.
• Each statement block may have a single or multiple statements to be executed. In case there is a single statement to be executed then it is not mandatory to enclose it in curly braces ({}) but if there are multiple statements then they must be enclosed in curly braces ({}).
• The else clause is optional and needs to be included only when some action is to be taken if the test condition evaluates to false.

Nested if . . . else

As opposed to the straightforward if statement, which may be used to test a single condition, these control structures are used to test for several conditions. The following is the syntax for nested if else:

Syntax:

```
if (conditional expression1)
{
statements1;
}
else if (conditional expression2)
{
statements2;
}
else if (conditional expression3)
{
statements3;
}
else
{
statements4;
}
```

Switch Statement

With the use of a range of character or numeric values, this selection statement enables us to evaluate the value of an expression. When a matching value is found, the control jumps to the statement related to that value, which is then run until the break statement or end of switch is reached. The syntax of the switch statement is as follows:

```
switch (Variable/Expression)
{
case Value1: statements1 ;
break ;
case Value2: statements2 ;
break ;
.
.
default:statements3 ;
}
```

Comparing Switch and If..else Statements

The if-else statement verifies both the logical expression and equality. Switch, on the other hand, just verifies equality. The if statement evaluates types such as boolean, character, pointer, floating-point, and integer. The switch statement, on the other hand, exclusively evaluates character or numeric datatypes.