Core Java 05101251

Prof. Priya Patel, Assistant Professor
Parul Institute of Computer Application

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-6

AWT Controls

Introduction to Applet

- Applets are small Java programs that are embedded in Web pages.
- They can be transported over the Internet from one computer (web server) to another (client computers).
- They transform web into rich media and support the delivery of applications via the Internet.

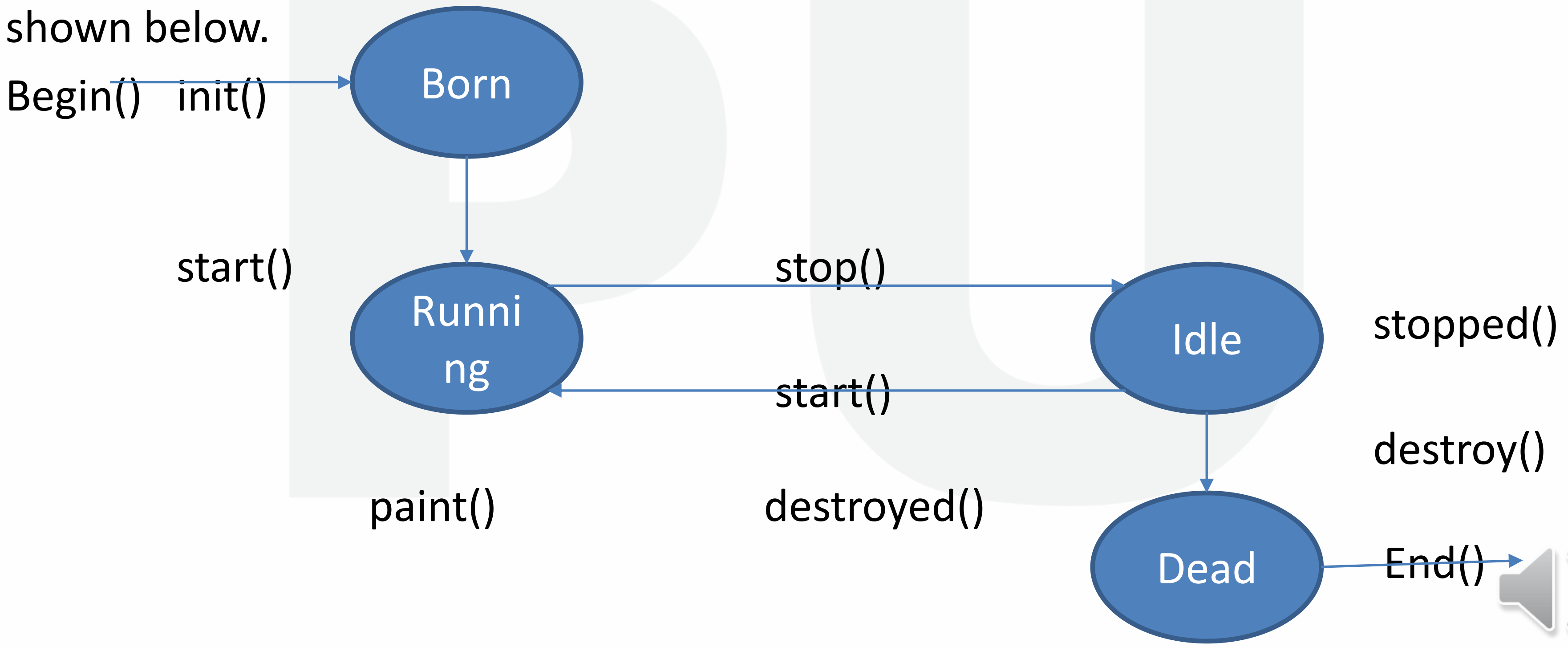
Introduction to Applet

- **How Applets differ from applications?**
- Although both the Applets and stand-alone applications are Java programs, there are certain restrictions are imposed on Applets due to security concerns:
- Applets don’t use the main() method, but when they are load, automatically call certain methods (init, start, paint, stop, destroy).
- They are embedded inside a web page and executed in browsers.
- They cannot read from or write to the files on local computer.
- They cannot communicate with other servers on the network.
- They cannot run any programs from the local computer.
- They are restricted from using libraries from other languages.

Life cycle

✓Every java applet inherits a set of default behaviours from the applet class.

✓As a result when applet is loaded it undergoes a series of changes in its state as shown below.



Life cycle

- The applet state includes:

1. Born or initialization state
2. Running state
3. Idle state
4. Dead or destroyed state

Born or Initialization state

- Applet enters the initialization state when it is first loaded.
- This is achieved by calling the init() method of applet class.
- The initialization occur only once in a applet life cycle.
- The applet is born, at this stage, we may do following things.
- Create object needed by the applet.
- Set up initial values.
- Load images or fonts.
- Set up colours.
- Syntax :

```
public void init()
{
        <ACTION>
```

Running State

•Applet enters into the running state when the system called start() of applet class.

•This occur automatically after the applet is initialized.

•Starting can also occur if the applet is already in stop state.

•Syntax :

```
public void start()
{
            <ACTION>
}
```

Idle State

- An applet becomes idle when it is stop from running.
- Stop occur automatically when we leave the page containing the running applet.
- We can also do same thing by calling stop method explicitly.
- Syntax :

```
public void stop()
{
            <ACTION>
}
```

Parul® University

DIGITAL LEARNING CONTENT

Display State

•Applet enters in display state whenever it has to perform some output operations on the screen.

•This happens immediately after the applet enter into running state.

•With the help of paint() we can display output on a screen.

•Almost every applet will have paint() method.

•Syntax :

```
public void paint(Graphics g)
{
        <ACTION>
}
```

Introduction to AWT

- Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.
- The java.awt package provides classes for AWT API such as Text Field, Label, Text Area, Radio Button, Check Box, Choice, List etc.

Introduction to AWT

- **Container :**

The Container is a component in AWT that can contain another components like buttons, text fields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

- **Window :**

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

- **Panel :**

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, text field etc.

Introduction to AWT

- **Frame** :

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, text field etc.

Methods of Component class

Method	Description
Public void add(Component c)	Insert a component.
Public void setSize(int width,int height)	Set the size of component
Public void setLayout(LayoutManager m)	Defines the layout manager for component.
Public void setVisible(Boolean status)	Changes the visibility of the component.by default it is false.

Component

- Component is the superclass of most of the displayable classes defined within the AWT. Note: it is abstract.
- It encapsulates all the attributes of visual component.
- Menu Component is another class which is similar to Component except it is the superclass for all GUI items which can be displayed within a drop-down menu.
- The Component class defines data and methods which are relevant to all Components

Component

1. setBounds
2. setSize
3. setLocation
4. setFont
5. setEnabled
6. setVisible
7. setForeground
8. setBackgroun

Container

- Container is a subclass of Component. (ie. All containers are themselves, Components)
- Containers contain components
- For a component to be placed on the screen, it must be placed within a Container
- The Container class defined all the data and methods necessary for managing groups of Components.

Container

```
Frame aFrame = new Frame("Hello World");
aFrame.setSize(100,100);
aFrame.setLocation(10,10);
aFrame.setVisible(true);
```

Panels

- When writing a GUI application, the GUI portion can become quite complex.
- To manage the complexity, GUIs are broken down into groups of components. Each group generally provides a unit of functionality.
- It is a concrete subclass of Container.
- A Panel is a rectangular Container whose sole purpose is to hold and manage components within a GUI.

DIGITAL LEARNING CONTENT

Panels

```
Panel aPanel = new Panel();
aPanel.add(new Button("Ok"));
aPanel.add(new Button("Cancel"));
Frame aFrame = new Frame("Button Test");
aFrame.setSize(100,100);
aFrame.setLocation(10,10);
aFrame.add(aPanel);
```

Label

- Label is a passive control because it does not create any event when accessed by the user.
- The label control is an object of Label. A label displays a single line of read-only text.
- However the text can be changed by the application programmer but cannot be changed by the end user in any way.

Button

- Button is a control component that has a label and generates an event when pressed.
- When a button is pressed and released, AWT sends an instance of ActionEvent to the button, by calling processEvent on the button.
- The button's processEvent method receives all events for the button; it passes an action event along by calling its own processActionEvent method.
- The latter method passes the action event on to any action listeners that have registered an interest in action events generated by this button.

Text Field

- When the user types a key in the text field the event is sent to the TextField.
- The key event may be key pressed, Key released or key typed.
- The key event is passed to the registered KeyListener.
- It is also possible to for an ActionEvent if the ActionEvent is enabled on the textfield then ActionEvent may be fired by pressing the return key.

List

- The List represents a list of text items.

Checkbox

- Checkbox control is used to turn an option on(true) or off(false).
- There is label for each checkbox representing what the checkbox does.
- The state of a checkbox can be changed by clicking on it.

Choice

- This class represents a dropdown list of Strings.
- Similar to a list in terms of functionality, but displayed differently.
- Only one item from the list can be selected at one time and the currently selected element is displayed.

Text Area

- This class displays multiple lines of optionally editable text.
- This class inherits several methods from TextComponent.
- Text Area also provides the methods:

1. appendText()
2. insertText()
3. replaceText()

Introduction to Swing

- Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

No	AWT	Swing
1	Platform-dependent	Platform-Independent
2	Heavyweight Component	Lightweight Component
3	Does not support pluggable look and feel	Support pluggable look and feel
4	Provides less component then swing	Provides more component then awt
5	Does not follow MVC	Does follow MVC

What is JFC

- The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications

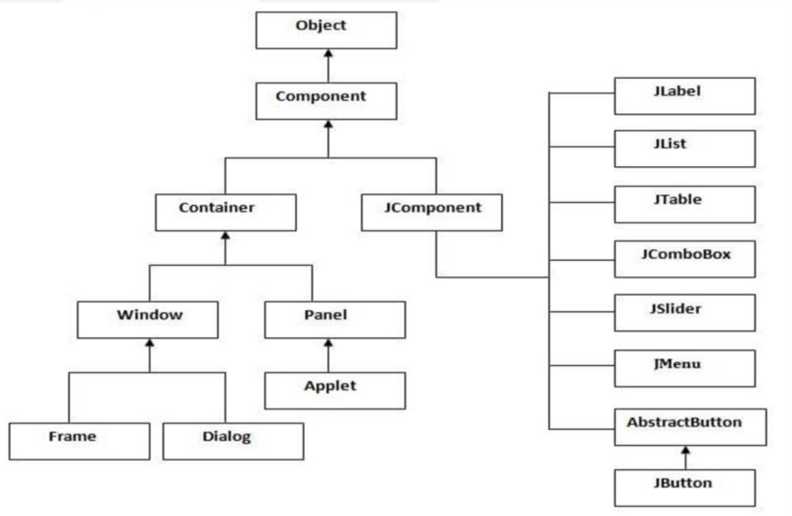


Image source : Google

Methods of Component class

Method	Description
Public void add(Component c)	Insert a component.
Public void setSize(int width,int height)	Set the size of component
Public void setLayout(LayoutManager m)	Defines the layout manager for component.
Public void setVisible(Boolean status)	Changes the visibility of the component.by default it is false.

JOptionPane

- Dialogs are windows that are more limited than frames.
- Every dialog is dependent on a frame.
- When that frame is destroyed, so are its dependent dialogs.
- When the frame is iconified, its dependent dialogs disappear from the screen.
- When the frame is decodified, its dependent dialogs return to the screen.
- To create simple dialogs, use the JOptionPane class.
- The dialogs that JOptionPane provides are *modal*.
- When a modal dialog is visible, it blocks user input to all other windows in the program.

Components

- **Containers**
- JFrame, JPanel, JApplet
- **Components**
- JButton, JTextField, JComboBox, JList, etc.
- **Helpers**
- Graphics, Color, Font, Dimension, etc.

What is an Event?

- Change in the state of an object is known as Event, i.e., event describes the change in the state of the source.
- Events are generated as a result of user interaction with the graphical user interface components.
- For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list, and scrolling the page are the activities that causes an event to occur.

Types of Event

- **Foreground Events :** These events require direct interaction of the user.
- They are generated as consequences of a person interacting with the graphical components in the Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page, etc.
- **Background Events :** These events require the interaction of the end user.
- Operating system interrupts, hardware or software failure, timer expiration, and operation completion are some examples of background events.

What is Event Handling?

- Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.
- This mechanism has a code which is known as an event handler, that is executed when an event occurs.
- Java uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.

What is Event Handling?

- The Delegation Event Model has the following key participants.

➢ **Source** :

The source is an object on which the event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide us with classes for the source object.

➢ **Listener** :

It is also known as event handler. The listener is responsible for generating a response to an event. From the point of view of Java implementation, the listener is also an object. The listener waits till it receives an event. Once the event is received, the listener processes the event and then returns.

What is Event Handling?

- The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event.
- The user interface element is able to delegate the processing of an event to a separate piece of code.
- In this model, the listener needs to be registered with the source object so that the listener can receive the event notification.
- This is an efficient way of handling the event because the event notifications are sent only to those listeners who want to receive them.

What is Event Handling?

➢**Steps Involved in Event Handling**

1. The user clicks the button and the event is generated.
2. The object of concerned event class is created automatically and information about the source and the event get populated within the same object.
3. Event object is forwarded to the method of the registered listener class.
4. The method is gets executed and returns.

The Event Class

- An event object has an event class as its reference data type.
- **The Event object class :** Defined in the java.util package.
- **The AWT Event class :** Defined in java.awt package.
- Root of all AWT based events.

Adapter Classes

- Java AWT Adapters are abstract classes from java.awt.event package introduced with JDK 1.1. We can create a listener class that extends its corresponding adapter class.
- Adapter classes make event handling simple and easy. Adapters usage replaces the listeners; adapters make event handling easy to the programmer.
- Every listener that includes more than one abstract method has got a corresponding adapter class.
- The advantage of adapter is that we can override any one or two methods we like instead of all.

Adapter Classes

1. ComponentAdapter
2. ContainerAdapter
3. FocusAdapter
4. KeyAdapter
5. MouseAdapter
6. MouseMotionAdapter
7. WindowAdapter

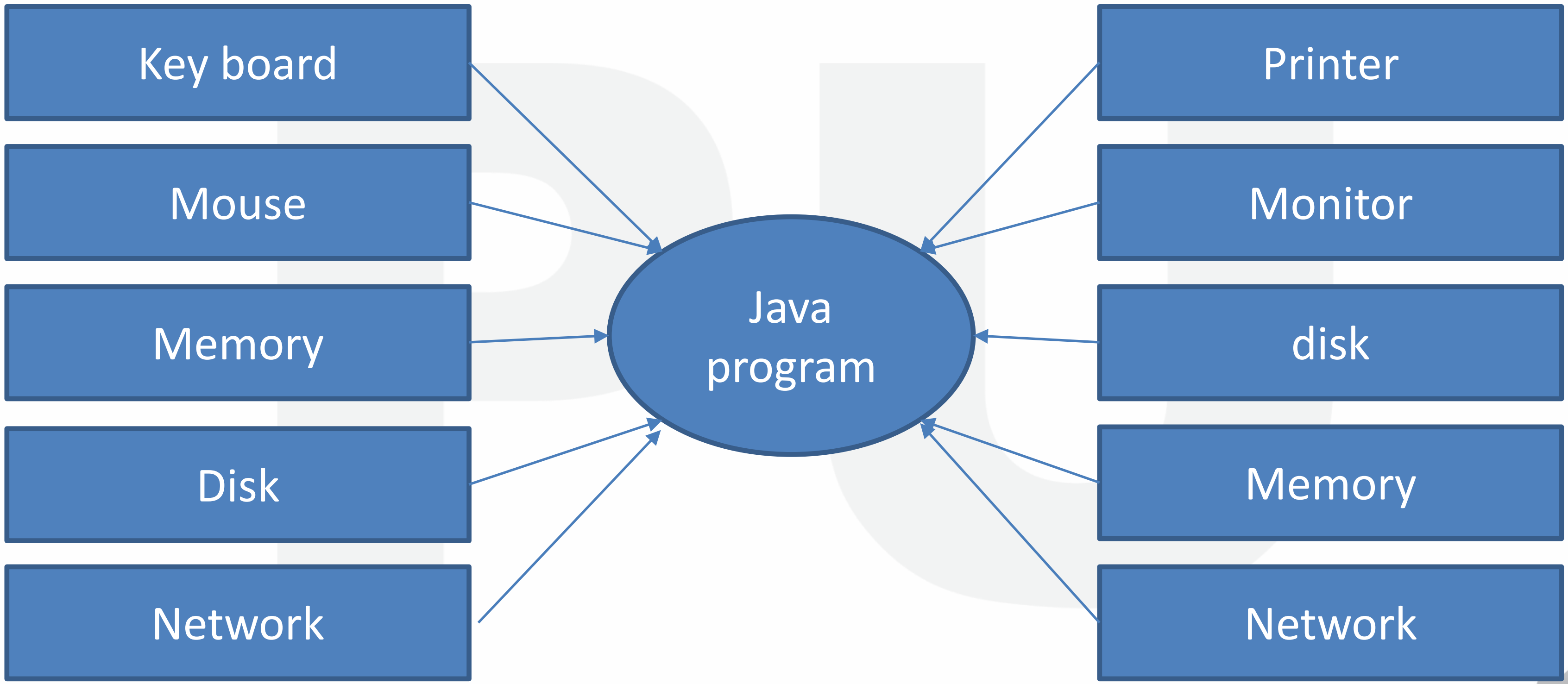
Concept of stream class

- It is a way of data flow from one source to another.
- In a file processing input refers to the flow of data into program and output means the flow of data out from a program.
- Input to a program may come from keyboard, module, memory, disk, network or another program.
- Output from a program may go to the stream, the printer, memory, disk and network or another program,

Parul® University

DIGITAL LEARNING CONTENT

Input - Output



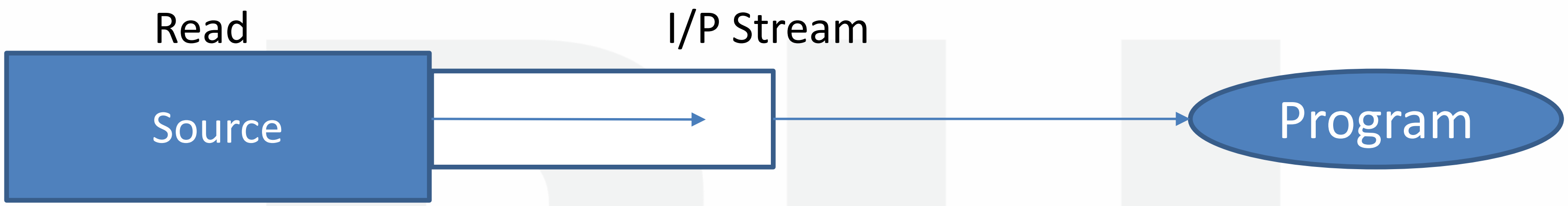
Classification of stream

- It is classified into two types.

1. InputStream
2. OutputStream

Classification of stream

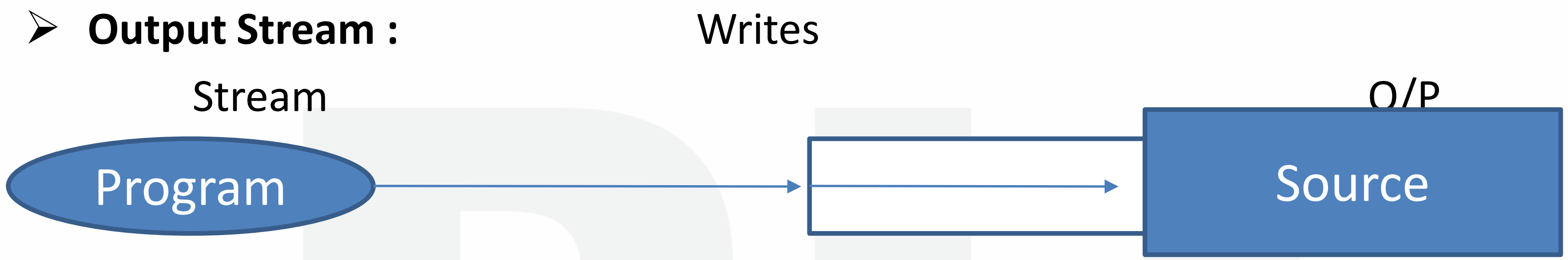
➤ **InputStream :**



- An input stream reads data from the source file and send it to the programmer.
- The program concepts and opens input stream and reads the data serially.

Classification of stream

- **Output Stream :**



- An output stream takes data from the program and writes it to the destination.
- The program concept opens output stream to the destination place of data and writes the data.

Classification of stream

- Streams are a clean way to deal with input/output without having every part of your code understand the difference between a keyboard and a network, for example.
- Java implements streams within class hierarchies defined in the java.io package.

1. Byte Stream Class : Handles i/o operation on byte.
2. Char Stream Class : Handles i/o operation on byte.

Classification of stream classes

➢ **Byte Stream Class :**

- Creates and manipulates stream and files for reading and writing bytes.
- Java streams are unidirectional.
- It can either read or write at a time.
- There are two byte stream class :

1. Input Stream Class
2. Output Stream Class

Byte Stream Class

➢Input Stream Class :

- Designed to read bytes from the files.
- It is an abstract class.
- The hierarchy of input stream class is shown in below diagram.

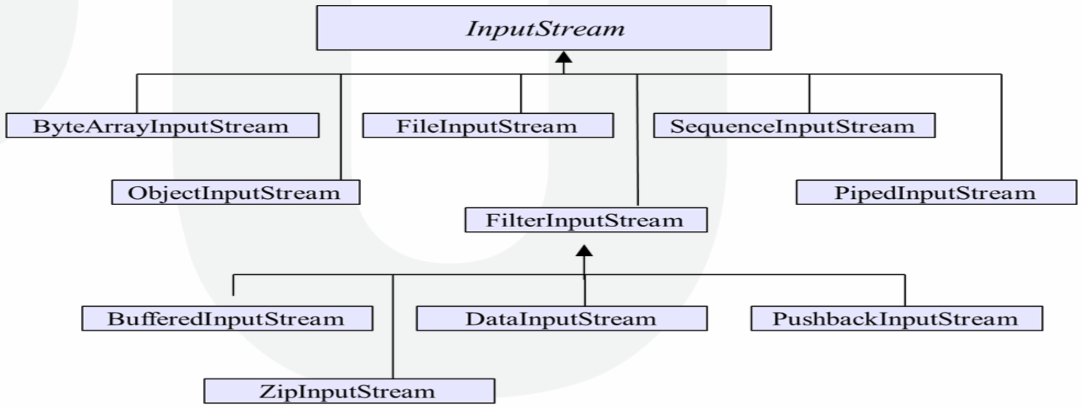


Image Source: Google

Parul® University

DIGITAL LEARNING CONTENT

Byte Stream Class

➢Input Stream Class define method for performing input functions like :

1. Reading bytes
2. Closing stream
3. Skipping
4. Marking position
5. Finding the no of bytes

Byte Stream Class

Methods	Description
Read()	Reads byte from the input stream
Read(byte b[])	Reads all bytes from array b
Read(byte b[],int n, int m)	Read m byte into b starting from nth byte
Available()	Gives no of bytes available
Skip(n)	Skip over n bytes
Reset()	Goes back to the beginning of the stream
Close()	Close the input stream

Byte Stream Class

- Data input stream implements on interface data input.
- Data Input interface methods are :

1. readInt()
2. readShort()
3. readLong()
4. readFloat()
5. readDouble()
6. readChar()
7. readBoolean()

Byte Stream Class

➢ Output Stream Class :

- Designed to write bytes into the files.
- It is an abstract class.
- The hierarchy of output stream class is shown in below diagram.

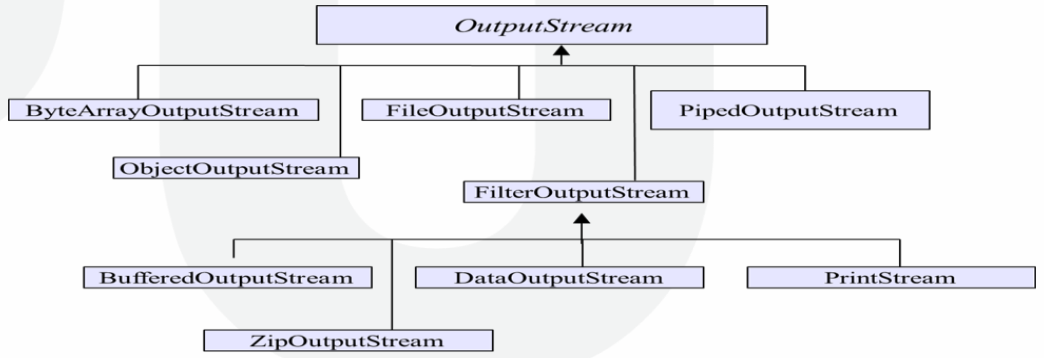


Image Source: Google

Byte Stream Class

- It performs the following operations :

1. Writing bytes
2. Closing stream
3. Flushing

Byte Stream Class

Method	Description
Write()	Write byte to the output stream
Write(byte b[])	Write all bytes from array b
Write(byte b[],int n, int m)	Write m byte from array b starting from nth byte
Close()	Close the output stream
Flush()	Flush the output stream

Byte Stream Class

- Data output stream implements on interface data input.
- Data Input interface methods are :

1. writeInt()
2. writeShort()
3. writeLong()
4. writeFloat()
5. writeDouble()
6. writeChar()
7. writeBoolean()

Character Stream Class

➢**Character Stream Class :**

- Creates and manipulates stream and files for reading and writing Characters.

Two character stream classes are there:

1. Reader Stream class
2. Writer Stream class

Character Stream Class

➢ **Reader Stream class :**

- Reads 16 bit Unicode characters from the files.

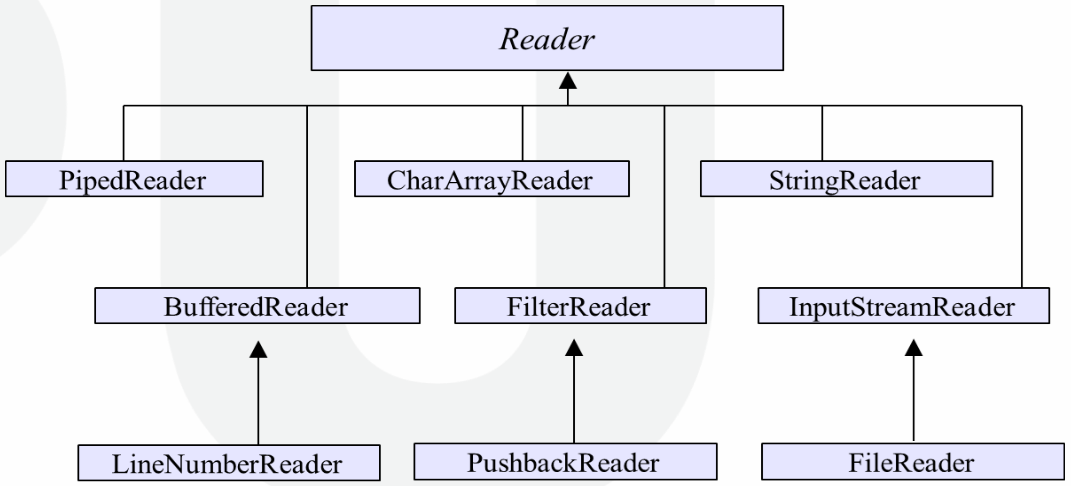


Image Source: Google

Character Stream Class

- Reader Stream Class define method for performing input functions like :

1. Reading characters
2. Closing stream
3. Skipping
4. Marking position
5. Finding the no of bytes

Character Stream Class

Methods	Description
Read()	Reads character from the reader stream
Read(byte b[])	Reads all bytes from array b to reader stream
Read(byte b[],int n, int m)	Read m character into b starting from nth character
Available()	Gives no of character available
Skip(n)	Skip over n character
Reset()	Goes back to the beginning of the stream
Close()	Close the input stream

Character Stream Class

- Reader class also provides methods:

1. readInt()
2. readShort()
3. readLong()
4. readFloat()
5. readDouble()
6. readChar()
7. readBoolean()

Character Stream Class

➢**Writer Stream class :**

- Designed to write character into the files.
- It is an abstract class.
- The hierarchy of writer stream class is shown in below diagram.

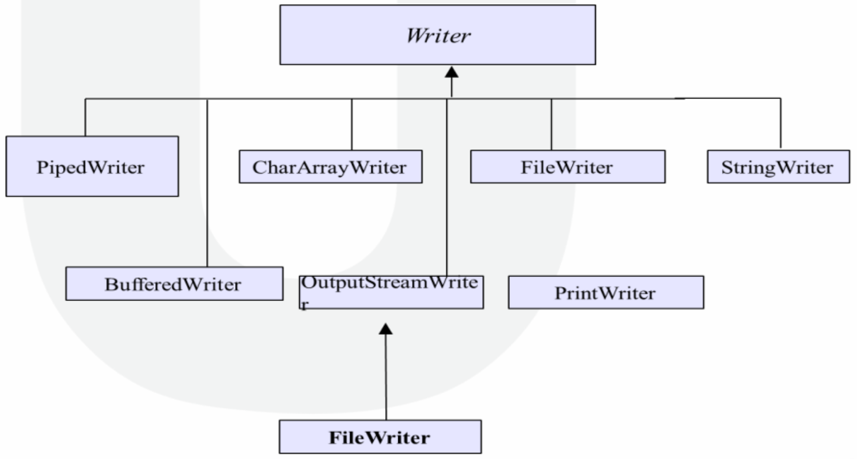


Image Source: Google

Character Stream Class

- It performs the following operations :

1. Writing character
2. Closing stream
3. Flushing

Parul® University

Character Stream Class

Methods	Description
Write()	Write character to the output stream
Write(byte b[])	Write all character from array b
Write(byte b[],int n, int m)	Write m characer from array b starting from nth character
Close()	Close the output stream
Flush()	Flush the output stream

What is File?

- Collection of related records on the disk
- Normally to store data in a program a variable or an array are used but the problems are :

1. The data is lost when the program is terminated or turn off the computer.
2. It is difficult to handle large amount of data
3. To solve the above problems the data are stored on secondary storage device using the file concept.

Operations on a file

1. Creating a file
2. Opening a file
3. Closing a file
4. Deleting a file
5. Getting name of a file
6. Getting size of a file
7. Checking the existence of a file
8. Renaming a file
9. Checking whether the file is readable
10. Checking whether the file is writable

Parul® University

DIGITAL LEARNING CONTENT

Reading and Writing character

- There are two subclasses for handling character in a file

1. FileReader : To read character from file
2. FileWriter : To write character into a file

Handling primitive data types

- The basis input and output streams provide read and write methods that can only be used for reading ,writing bytes and character.
- There are two classes :

1. DataInputStream
2. DataOutputStream

Definitions

- **Persistent data** : A data stored in a file is known as persistent data
- **File Processing** : A process of storing and managing a file
- **Object Serialization** : The process of reading and writing object
- **Stream** : The way of data flow from source to another

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in