Unit 1

Web Technology

Index/ Content

Index/ Content

Index/ Content

Index/ Content

Prerequisite and Recap

- Internet Fundamentals
- Web Fundamentals
- Web Design Fundamentals

Introduction to Web Technologies

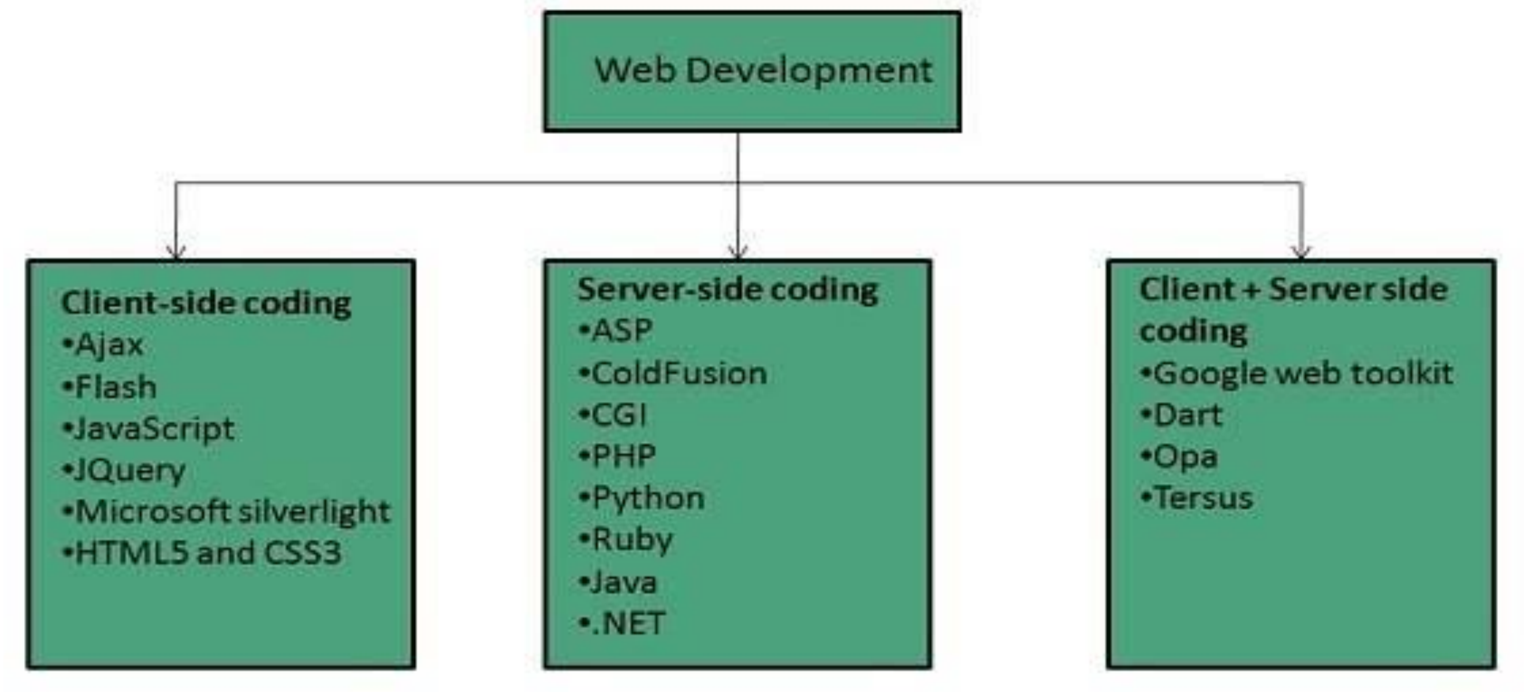
- The methods by which computers communicate with each other through the use of markup languages and multimedia packages is known as **web technology**.
- In the past few decades, web technology has undergone a dramatic transition, from a few marked up web pages to the ability to do very specific work on a network without interruption.
- It involves the use of hypertext markup language (HTML) and cascading style sheets (CSS), JavaScript etc.

Web Development Strategies

It is not enough to just have a website anymore these days. You have to have a strategy implemented to get the best results that you can have from a website. The first thing that you will have to think about is the name of your url and how to pick a good name that people are searching for. The next task in your strategy should be to pick a good server that will be able to handle your website visitors and the content that you serve on your website. The third thing that you need to choose is a CMS or content management system that will help you prepare your content for the web. Yet there is a lot more to web development strategy than this. These are just the basics to get you started.

Web Development Strategies

- **Web development** refers to building website and deploying on the web. Web development requires use of scripting languages both at the server end as well as at client end.



Web Development Strategies

Before developing a web site one should keep several aspects in mind like:

•What to put on the web site?

•Who will host it?

•How to make it interactive?

•How to code it?

•How to create search engine friendly web site?

•How to secure the source code frequently?

•Will the web site design display well in different browsers?

•Will the navigation menus be easy to use?

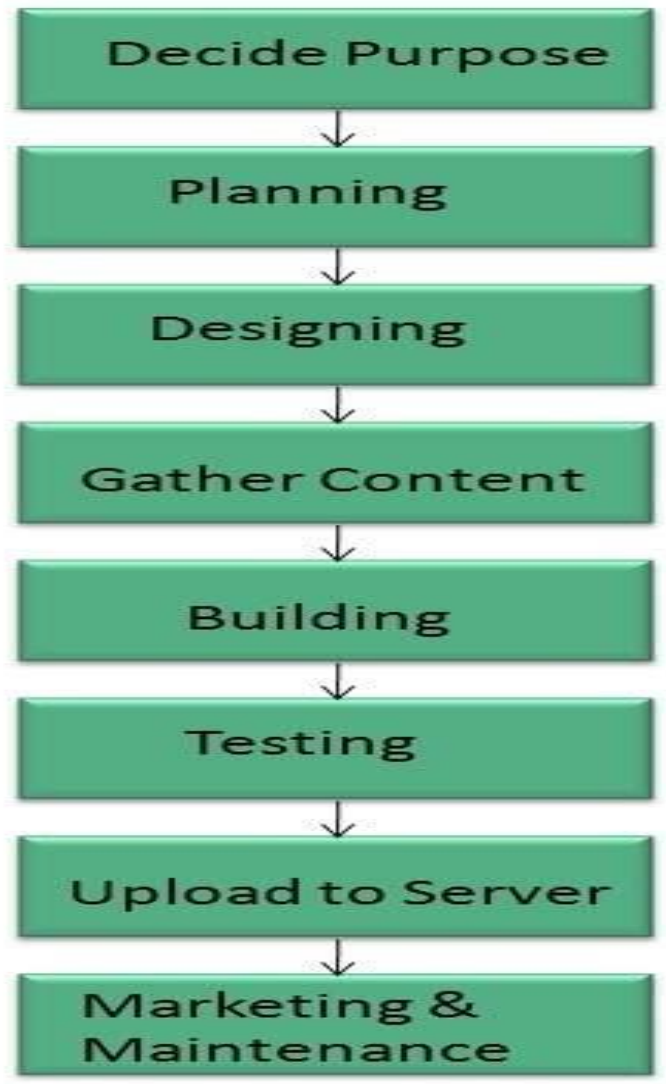
•Will the web site loads quickly?

•How easily will the site pages print?

•How easily will visitors find important details specific to the web site?

•How effectively the style sheets be used on your web sites?

Web Development Process



History of Web and Internet

- Sir Tim Berners-Lee is a British computer scientist. By October of 1990, Tim had written the three fundamental technologies that remain the foundation of today’s web (and which we may have seen appear on parts of our web browser):
    - HTML: HyperText Markup Language. The markup (formatting) language for the web.
    - URI: Uniform Resource Identifier. A kind of “address” that is unique and used to identify to each resource on the web. It is also commonly called a URL.
    - HTTP: Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the web.

Protocols governing Web

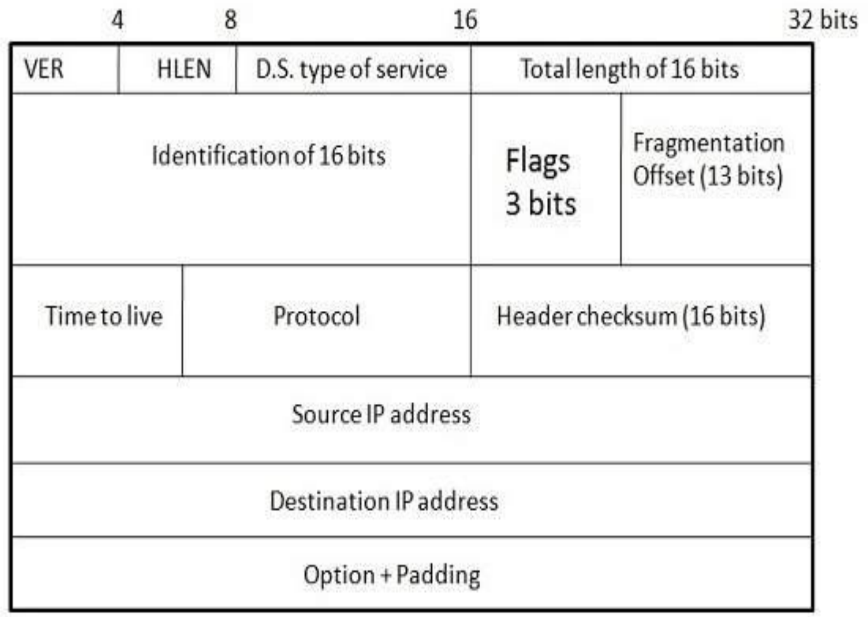
- A Protocols is a set of rules. Protocols allows two computers to communicate over media such as wireless or hardwired technologies.
- When computers communicate with each other, there needs to be a common set of rules and instruction that each computer follows. Protocols are the language of computers.

Transmission Control Protocol (TCP)

- TCP is a connection oriented protocol and offers end-to-end packet delivery. It acts as back bone for connection. It exhibits the following key features:
- Transmission Control Protocol (TCP) corresponds to the Transport Layer of OSI Model.
- TCP is a reliable and connection oriented protocol.
- TCP offers:
    - Stream Data Transfer.
    - Reliability.
    - Efficient Flow Control
    - Full-duplex operation.
    - Multiplexing.
- TCP offers connection oriented end-to-end packet delivery.
- TCP ensures reliability by sequencing bytes with a forwarding acknowledgement number that indicates to the destination the next byte the source expect to receive.
- It retransmits the bytes not acknowledged with in specified time period.

Internet Protocol (IP)

- Internet Protocol is **connectionless** and **unreliable** protocol. It ensures no guarantee of successfully transmission of data.
- In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.
- Internet protocol transmits the data in form of a datagram as shown in the following diagram



The length of datagram is variable.

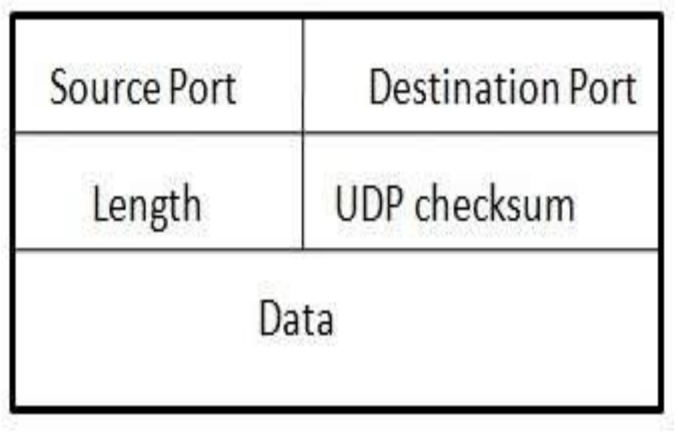
The Datagram is divided into two parts: **header** and **data.**

The length of header is 20 to 60 bytes.

The header contains information for routing and delivery of the packet

User Datagram Protocol (UDP)

Like IP, UDP is connectionless and unreliable protocol. It doesn’t require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received.
UDP transmits the data in form of a datagram. The UDP datagram consists of five parts as shown in the following diagram

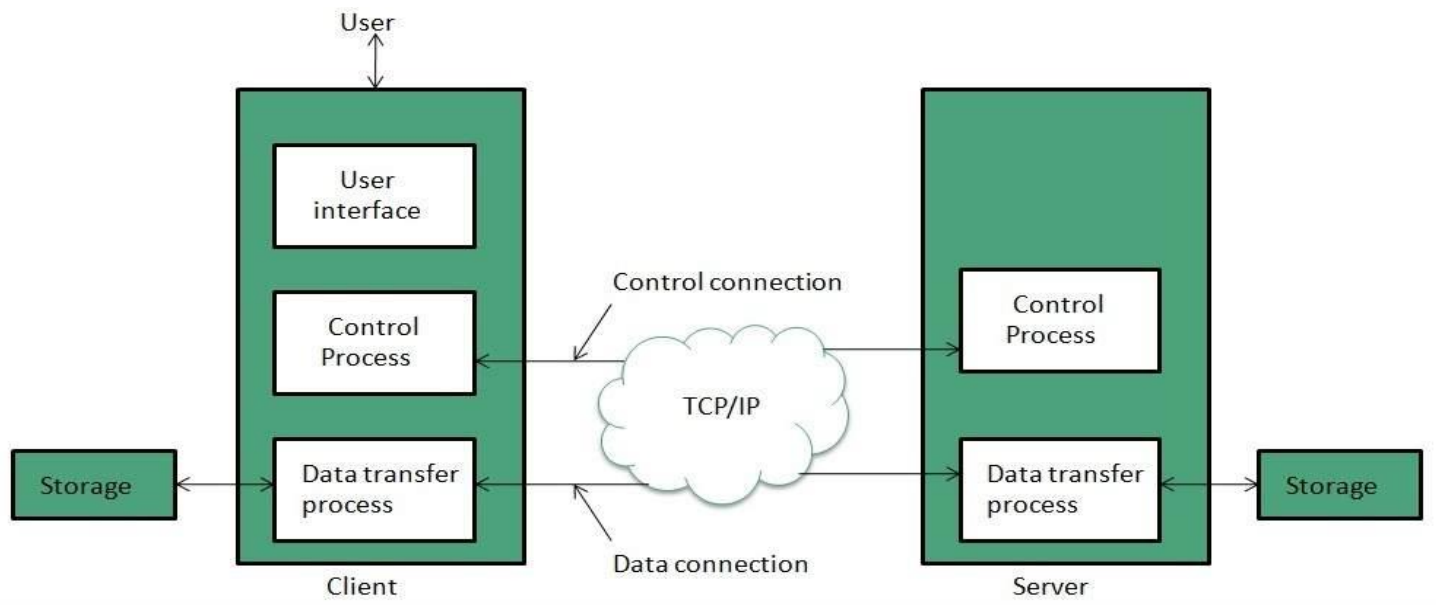


Source Port	Destination Port
Length	UDP checksum
Data	

UDP is used by the application that typically transmit small amount of data at one time. UDP provides protocol port used i.e. UDP message contains both source and destination port number, that makes it possible for UDP software at the destination to deliver the message to correct application program.

File Transfer Protocol (FTP)

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:
FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
FTP establishes two different connections: one is for data transfer and other is for control information.
FTP uses **port 21** for the control connection and **Port 20** for the data connection.



Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol is also used to transfer the files but it transfers the files without authentication. Unlike FTP, TFTP does not separate control and data information. Since there is no authentication exists, TFTP lacks in security features therefore it is not recommended to use TFTP.

Key points

TFTP makes use of UDP for data transport. Each TFTP message is carried in separate UDP datagram.

The first two bytes of a TFTP message specify the type of message.

The TFTP session is initiated when a TFTP client sends a request to upload or download a file.

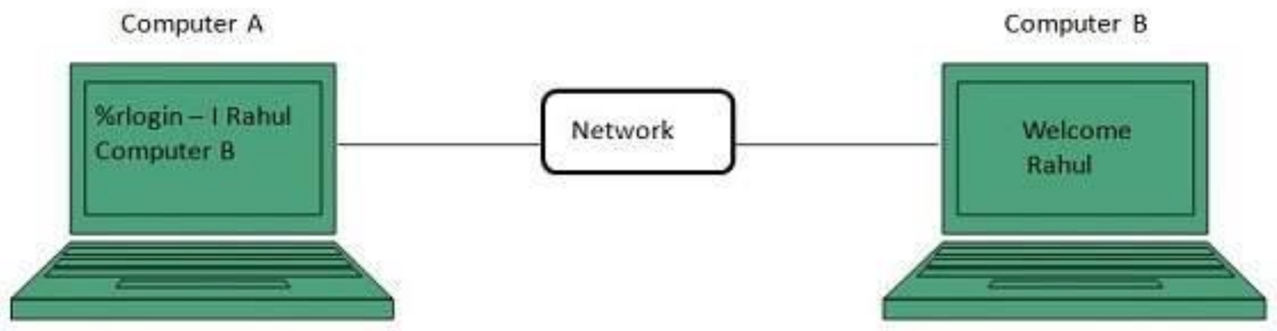
The request is sent from an ephemeral UDP port to the **UDP port 69** of an TFTP server.

Difference between FTP and TFTP

S.N.	Parameter	FTP	TFTP
1	Operation	Transferring Files	Transferring Files
2	Authentication	Yes	No
3	Protocol	TCP	UDP
4	Ports	21 – Control, 20 – Data	Port 3214, 69, 4012
5	Control and Data	Separated	Separated
6	Data Transfer	Reliable	Unreliable

Telnet

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface. The following diagram shows a person is logged in to computer A, and from there, he remote logged into computer B.



Hyper Text Transfer Protocol (HTTP)

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs.

HTTP request comprises of lines which contains:

- Request line
- Header Fields
- Message body

Key Points

- The first line i.e. the **Request line** specifies the request method i.e. **Get** or **Post.**
- The second line specifies the header which indicates the domain name of the server from where index.htm is retrieved.

HTTP response also has certain structure. HTTP response contains:

- Status line
- Headers
- Message body

Writing Web Projects

- Developing web project is a crucial activity and web project development differs from traditional web projects.
- **Steps of writing a web projects are:**
  - **Write a project mission statement:** Write the specific mission statement that we want to do.
  - **Identify Objectives**:
    - Specific
    - Measurable
    - Attainable
    - Realistic
    - Time limited

Writing Web Projects

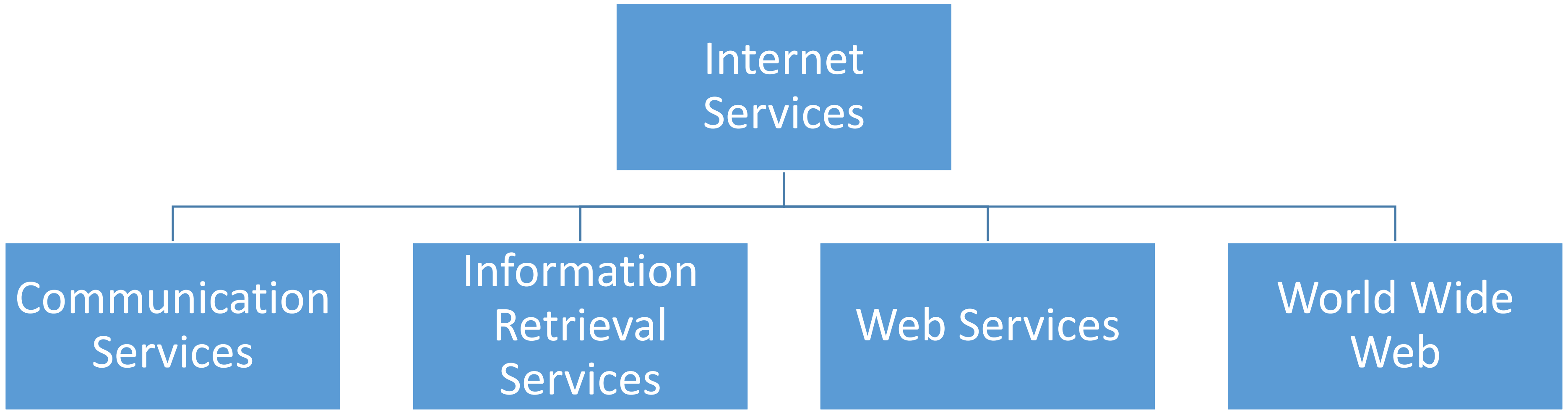
- **Identify our target users:** The matter of a website will be determined by the users whom we want to visit the site. This is totally depend upon.
  - Market research
  - Focus group
  - Understanding intranet audiences
- **Determine the scope:** By supporting documents and client's approval.
- **Budget**
  - Assumption for budgets.
  - Budget categories.
  - Determine hidden costs and tools.
- **Planning issues**
  - Discuss client's existing information system.
  - Project team and developing infrastructure.
  - Where the website will place.

Connecting to Internet

- Depending on whether we have a cable or DSL internet service provider (ISP), the steps we will need to take to setup our internet connection will be different.
  1. Setting up cable internet connection.
  2. Setting up Digital subscriber line (DSL) internet connection.

Introduction to Internet services and tools

- Internet Services allows us to access huge amount of information such as text, graphics, sound and software over the internet.
- Following diagram shows the four different categories of Internet Services.

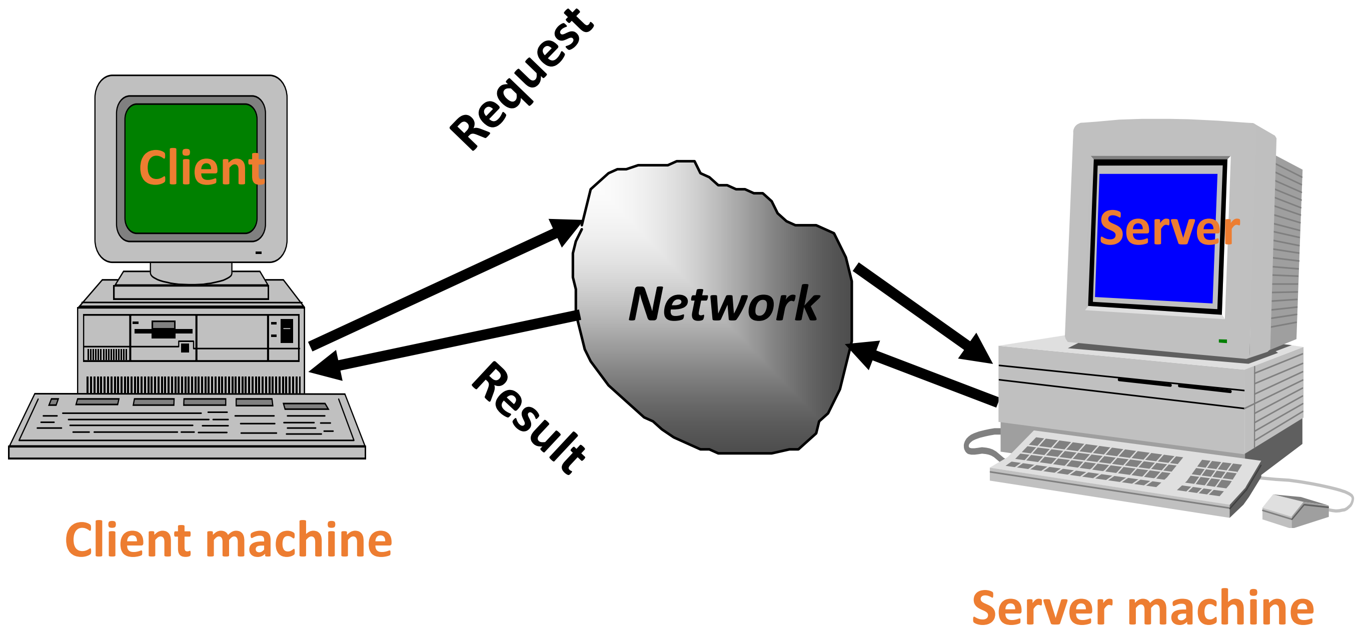


Communication Services

S. No	Service Description
1	Electronic Mail - Used to send electronic message over the internet.
2	Telnet - Used to log on to a remote computer that is attached to internet.
3	Newsgroup - Offers a forum for people to discuss topics of common interests.
4	Internet Relay Chat (IRC) - Allows the people from all over the world to communicate in real time.
5	Mailing Lists - Used to organize group of internet users to share common information through e-mail.
6	Internet Telephony (VoIP) - Allows the internet users to talk across internet to any PC equipped to receive the call.
7	Instant Messaging - Offers real time chat between individuals and group of people. Eg. Yahoo messenger, MSN messenger.

Introduction to client-server computing

- A server software accepts requests for data from client software and returns the results to the client is called as Client-Server (CS) Computing.



Characteristics of Client Server Computing

- The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.
- The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.
- A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.
- Denial of Service attacks servers ability to respond to authentic client requests by inundating it with false requests.
- An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

Advantages of Client Server Computing

- All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorization and authentication.
- The server need not be located physically close to the clients. Yet the data can be accessed efficiently.
- It is easy to replace, upgrade or relocate the nodes in the client server model because all the nodes are independent and request data only from the server.
- All the nodes i.e. clients and server may not be build on similar platforms yet they can easily facilitate the transfer of data.

Disadvantages of Client Server Computing

- If all the clients simultaneously request data from the server, it may get overloaded. This may lead to congestion in the network.
- If the server fails for any reason, then none of the requests of the clients can be fulfilled. This leads of failure of the client server network.
- The cost of setting and maintaining a client server model are quite high.

What is Core Java

The word ***Core*** describes the basic concept of something, and here, the phrase *'Core Java'* defines the basic Java that covers the basic concept of Java programming language. We all are aware that Java is one of the well-known and widely used programming languages, and to begin with it, the beginner has to start the journey with Core Java and then towards the Advance Java. The Java programming language is a general-purpose programming language that is based on the OOPs concept

The principle followed by Java is **WORA** that says *Write Once, Run Anywhere*. The programming language is quite simple and easy to understand.

Java OOPs Concepts

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object

Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Class

Collection of objects is called class. It is a logical entity.
A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

- If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- In Java, we use method overloading and method overriding to achieve polymorphism.
- Another example can be to speak something; for example, a cat speaks meow, dog barks etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Operators, Data Types, Variables & Flow Control in Java

- Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Operators precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Unary Operator

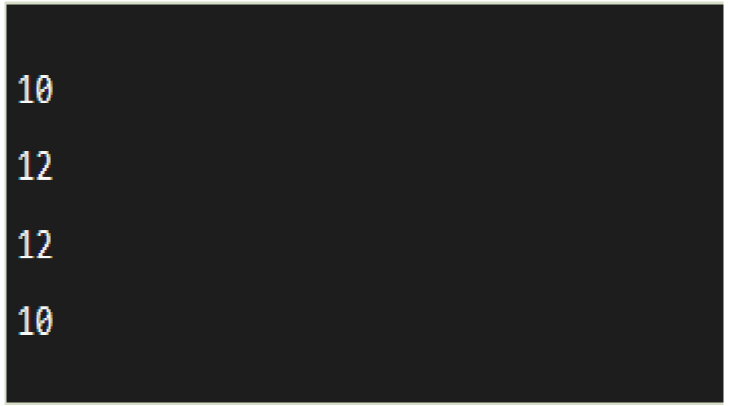
The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Unary Operator Example: ++ and --

```
public class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++);//10 (11)
System.out.println(++x);//12
System.out.println(x--);//12 (11)
System.out.println(--x);//10
}}
```

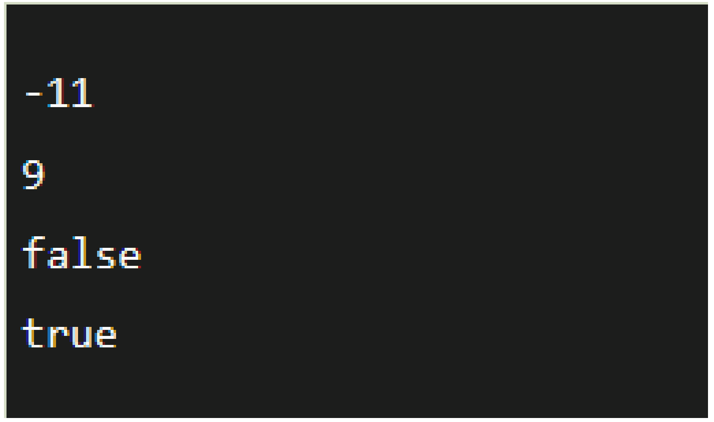
Output:



```
10
12
12
10
```

Java Unary Operator Example: ~ and !

Output:


```
-11

9

false

true
```

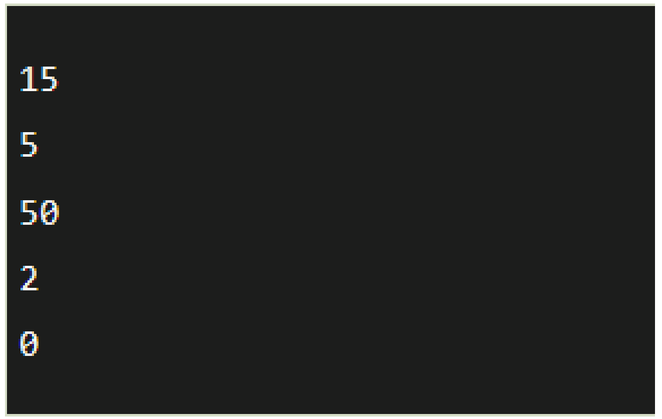
```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=-10;
boolean c=true;
boolean d=false;
System.out.println(~a);//-
11 (minus of total positive value which starts from 0)
System.out.println(~b);//9 (positive of total minus, positive starts from 0)
System.out.println(!c);//false (opposite of boolean value)
System.out.println(!d);//true
}}
```

Java Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}}
```

Output:



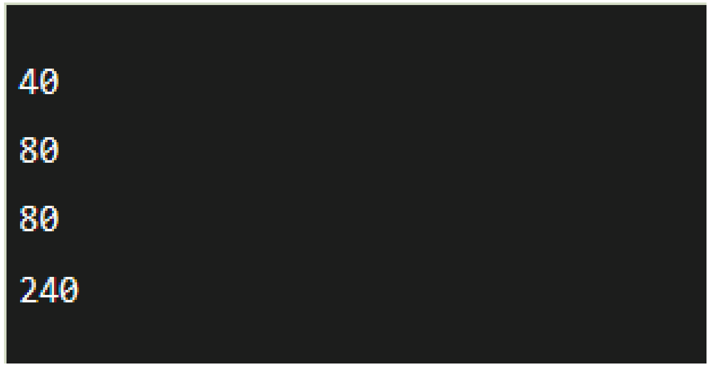
```
15
5
50
2
0
```

Java Left Shift Operator

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times

```
public class OperatorExample{
public static void main(String args[]){
System.out.println(10<<2);//10*2^2=10*4=40
System.out.println(10<<3);//10*2^3=10*8=80
System.out.println(20<<2);//20*2^2=20*4=80
System.out.println(15<<4);//15*2^4=15*16=24
0
}}
```

Output:



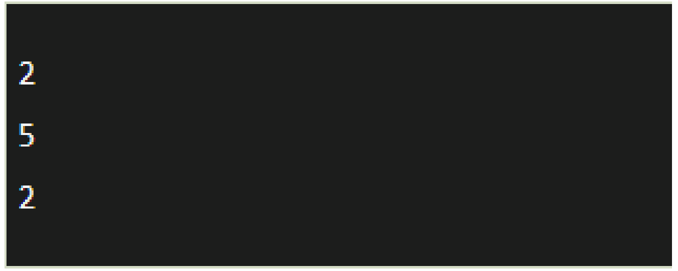
```
40
80
80
240
```

Java Right Shift Operator

The Java right shift operator >> is used to move the value of the left operand to right by the number of bits specified by the right operand.

```
public OperatorExample{
public static void main(String args[]){
System.out.println(10>>2);//10/2^2=10/4=2

System.out.println(20>>2);//20/2^2=20/4=5

System.out.println(20>>3);//20/2^3=20/8=2

}}
```

Output:



```
2
5
2
```

Java AND Operator Example: Logical && and Bitwise &

The logical && operator doesn't check the second condition if the first condition is false. It checks the second condition only if the first one is true.
The bitwise & operator always checks both conditions whether first condition is true or false

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a<b&&a<c);//false && true = false
System.out.println(a<b&a<c);//false & true = false
}}
```

```
false
false
```

Java OR Operator Example: Logical || and Bitwise |

The logical || operator doesn't check the second condition if the first condition is true. It checks the second condition only if the first one is false.
The bitwise | operator always checks both conditions whether first condition is true or false.

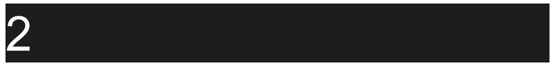
```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a>b||a<c);//true || true = true
System.out.println(a>b|a<c);//true | true = true
//|| vs |
System.out.println(a>b||a++<c);//true || true = true
System.out.println(a);//10 because second condition is not checked

System.out.println(a>b|a++<c);//true | true = true
System.out.println(a);//11 because second condition is checked
}}
```

```
True
True
True
10
True
11
```

Java Ternary Operator

Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

```
public class OperatorExample{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}}
```



```
2
```

Java Assignment Operator

Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

```
public class OperatorExample{
public static void main(String args[]){
int a=10;
int b=20;
a+=4;//a=a+4 (a=10+4)
b-=4;//b=b-4 (b=20-4)
System.out.println(a);
System.out.println(b);
}}
```

```
14
16
```

Expressions and data types

- An expression is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

```
int cadence = 0;
anArray[0] = 100;
System.out.println("Element 1 at index 0: " + anArray[0]);
```

- There are two data types available in Java
  - Primitive Data Types
  - Non-Primitive/Reference/Object Data Types

Primitive Data Types

Type	Size	Range	Default Value	Examples
Boolean	1 bit	NA	false	boolean bool = true;
char	16 bits	Unicode Characters	‘\u0000’ or 0, which is nothing but a white space	char c = ‘A’; char c = ‘\u0041’; char c = 65; char c = ‘\t’;
byte	8 bits	[-128 - 127] or [-2^7 to 2^7-1]	0	byte b = 10; byte b = 0b010;
short	16 bits	[-32768 - 32767]	0	short s = 32; short s = ‘A’;

Primitive Data Types

Type	Size	Range	Default Value	Examples
int	32 bits	[-2147483648 - 2147483647]	0	int i = 10; int i = ‘A’;
long	64 bits	[-2^63 - 2^63-1]	0	long l = 3200L; long l = 3200;
float	32 bits	[-3.4E38 - 3.4E38]	0.0f	float f = (float) 12.34; float f = 12.34f;
double	64 bits	[-1.7E308 - 1.7E308]	0.0	double d = 12.34;

Variables

The variable is the basic unit of storage in a Java program. A variable is defined by the combination of an identifier, a type, and an optional initializer. In addition, all variables have a scope, which defines their visibility, and a lifetime.

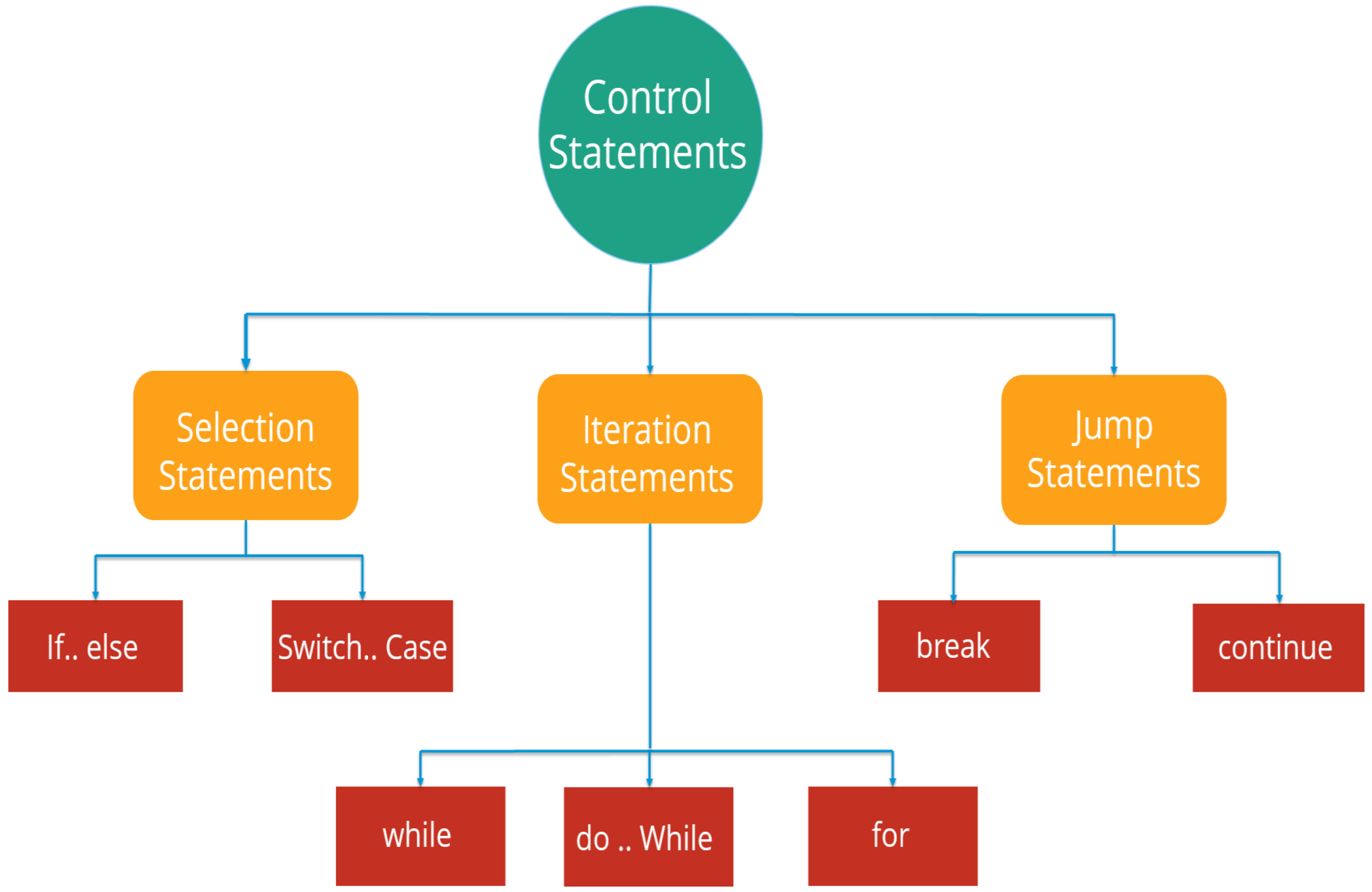
Example:

```
int a, b, c;
int x=10,y, z=90;
char ch='A';
Boolean bl=true;
```

Control Flow Statements

- The statements inside our source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling our program to conditionally execute particular blocks of code. There are three types of control statements:
  - The decision-making statements (if-then, if-then-else, switch)
  - The looping statements (for, while, do-while)
  - The branching statements (break, continue, return)

Control Flow Statements



If Statements

In Java, the "if" statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition. The condition of the If statement gives a Boolean value, either true or false. In Java, there are four types of if-statements given below

- Simple if statement
- if-else statement
- if-else-if ladder
- Nested if-statement

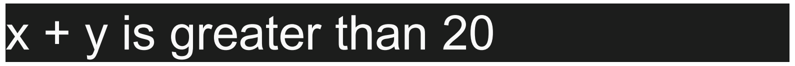
Simple if statement:

It is the most basic statement among all control flow statements in Java. It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.
Syntax of if statement is given below.

```
if(condition) {
statement 1; //executes when condition is true
}
```

Student.java

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y > 20) {
System.out.println("x + y is greater than 20");
  }   }     }
```



```
x + y is greater than 20
```

if-else statement

The if-else statement
is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

Syntax:

```
if(condition) {
statement 1; //executes when condition
is true
}
else{
statement 2; //executes when condition
 is false
}
```

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y < 10) {
System.out.println("x + y is less than      10");
}   else {
System.out.println("x + y is greater than 20");
}
}
}
```

```
x + y is greater than 20
```

if-else-if ladder:

The if-else-if statement contains the if-statement followed by multiple else-if statements. In other words, we can say that it is the chain of if-else statements that create a decision tree where the program may enter in the block of code where the condition is true. We can also define an else statement at the end of the chain.
Syntax of if-else-if statement is given below.

```
if(condition 1) {
statement 1; //executes wh
en condition 1 is true
}
else if(condition 2) {
statement 2; //executes wh
en condition 2 is true
}
else {
statement 2; //executes wh
en all the conditions are fals
e
}
```

Delhi

Student.java

```
public class Student {
public static void main(String[] args) {
String city = "Delhi";
if(city == "Meerut") {
System.out.println("city is meerut");
}else if (city == "Noida") {
System.out.println("city is noida");
}else if(city == "Agra") {
System.out.println("city is agra");
}else {
System.out.println(city);
} } }
```

Nested if-statement

In nested if-statements, the if statement can contain a **if** or **if-else** statement inside another if or else-if statement.
Syntax of Nested if-statement is given below.

```
if(condition 1) {
statement 1; //executes wh
en condition 1 is true
if(condition 2) {
statement 2; //executes wh
en condition 2 is true
}
else{
statement 2; //executes wh
en condition 2 is false
}
}
```

```
Delhi
```

```
public class Student {
public static void main(String[] args) {
String address = "Delhi, India";

if(address.endsWith("India")) {
if(address.contains("Meerut")) {
System.out.println("Your city is Meerut");
}else if(address.contains("Noida")) {
System.out.println("Your city is Noida");
}else {
System.out.println(address.split(",")[0]);
}
}else {
System.out.println("You are not living in India");
}   }   }
```

Switch Statement

In Java, Switch statements
are similar to if-else-if statements. The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched. The switch statement is easier to use instead of if-else-if statements. It also enhances the readability of the program.

Points to be noted about switch statement:

- The case variables can be int, short, byte, char, or enumeration. String type is also supported since version 7 of Java
- Cases cannot be duplicate
- Default statement is executed when any of the case doesn't match the value of expression. It is optional.
- Break statement terminates the switch block when the condition is satisfied.
  It is optional, if not used, next case is executed.
- While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

Syntax for Switch

```
switch (expression)
{
   case value1:
    statement1;
    break;
   .
   .
   case valueN:
    statementN;
    break;
   default:
    default statement;
}
```

Example

```
public class Student implements Cloneable {
public static void main(String[] args) {
int num = 2;
switch (num){
case 0:
System.out.println("number is 0");
break;
case 1:
System.out.println("number is 1");
break;
default:
System.out.println(num);
} } }
```

```
2
```

Loop Statements

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

In Java, we have three types of loops that execute similarly. However, there are differences in their syntax and condition checking time.

- for loop
- while loop
- do-while loop

Java for loop

In Java, for loop is similar to C and C++.
It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.

Syntax

```
for(initialization, condition,
 increment/decrement)
{
//block of statements
}
```

Calculation.java

```
public class Calculattion {
public static void main(String[] args) {
// TODO Auto-generated method stub
int sum = 0;
for(int j = 1; j<=10; j++) {
sum = sum + j;
}
System.out.println("The sum of first 10 natural
 numbers is " + sum);
} }
```

```
The sum of first 10 natural numbers is 55
```

Java for-each loop

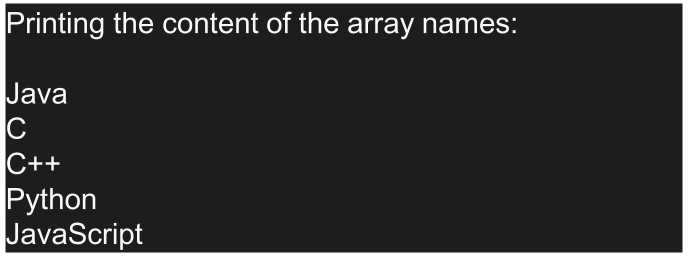
Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable. The syntax to use the for-each loop in java is given below.

Syntax:

```
for(data_type var :
array_name/collec
tion_name){
//statements
}
```

Calculation.java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
String[] names = {"Java","C","C++","Python","JavaScript"};
System.out.println("Printing the content of the array names:\n"
);
for(String name:names) {
System.out.println(name);
}   }   }
```



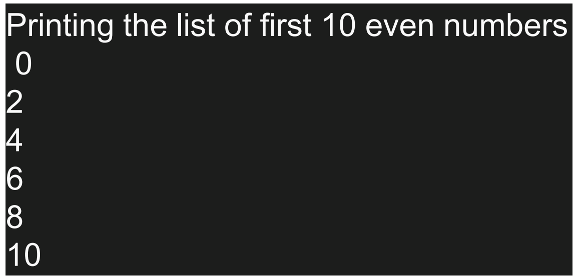
```
Printing the content of the array names:

Java
C
C++
Python
JavaScript
```

Java while loop

The while loop is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization and increment/decrement doesn't take place inside the loop statement in while loop.
It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

Syntax

```
while(condition){
//looping statements

}
```



```
Printing the list of first 10 even numbers
 0
2
4
6
8
10
```

Calculation.java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbe
rs \n");
while(i<=10) {
System.out.println(i);
i = i + 2;
}   }   }
```

Java do-while loop

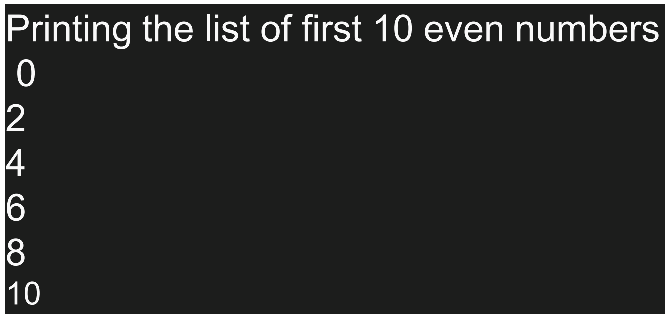
The do-while loop checks the condition at the end of the loop after executing the loop statements. When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.
It is also known as the exit-controlled loop since the condition is not checked in advance.

Syntax

```
do
{
//statements
} while (condition);
```

Calculation.java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n")
;
do {
System.out.println(i);
i = i + 2;
}while(i<=10);
}    }
```



```
Printing the list of first 10 even numbers
 0
2
4
6
8
10
```

Jump Statements

Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program. There are two types of jump statements in Java, i.e., break and continue.

Java break statement

As the name suggests, the break statement is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop.
The break statement cannot be used independently in the Java program, i.e., it can only be written inside the loop or switch statement.

Breakexample.java

```
public class BreakExample {

public static void main(String[] args) {
for(int i = 0; i<= 10; i++) {
System.out.println(i);
if(i==6) {
break;
}
}
}
}
```

```
0
1
2
3
4
5
6
```

Java continue statement

Unlike break statement, the continue statement doesn't break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately.
Consider the following example to understand the functioning of the continue statement in Java.

```
public class ContinueExample {

public static void main(String[] args) {

for(int i = 0; i<= 2; i++) {

for (int j = i; j<=5; j++) {

if(j == 4) {
continue;  }
System.out.println(j);  }  }  }  }
```

```
0
1
2
3
5
1
2
3
5
2
3
5
```

Array in Java

- Arrays in Java is similar to that of C++ or any other programming language.
- An array is an container object or data structure, which stores a fixed size sequential collection of elements of the same types.
- The length of an array is established when the array is created. After creation, its length is fixed.
- Each variable in the array is called an array element.
- To reference a particular element in an array, we use the array name combined with an integer value of type int, called an index.
- The first element has an index of 0 always.
- There are two types of arrays in Java:
  Single-dimension Array
  Multi-dimension Array

Declaring a Variable to Refer to an Array

- The preceding program declares an array (named anArray) with the following line of code:
  - **int[] anArray;**
  - **int anArray[];**

- Create an array is with the **new** operator.
  - anArray = new int[10];

- Element by Element initialization of an Array
  - anArray[0] = 100;
  - anArray[1] = 200;
  - anArray[2] = 300;

- Another way of initialize an array:
  - int[] anArray = {100, 200, 300,400, 500, 600, 700, 800, 900, 1000};

The Length of an Array

- We can refer to the length of the array — the number of elements it contains — using length, a data member of the array object.
- For example, for the array myArray, we can refer to its length as myArray.length.

- We can use the length member of an array to control a numerical for loop that iterates over the elements of an array.

Multidimensional Arrays

- Multidimensional arrays are actually arrays of arrays.
- To declare a multidimensional array variable, specify each additional index using another set of square brackets.

- Example:
    - int twoD[ ][ ] = new int[4][5] ;

String

- String is a sequence of characters. But in Java, a string is an object that represents a sequence of characters.
- The java.lang.String class is used to create string object.
- There are two ways to create a String object:
  - **By string literal** : Java String literal is created by using double quotes.
    - For Example: String s="Welcome";
  - **By new keyword** : Java String is created by using a keyword "new".
    - For example: String s=new String("Welcome");
  - It creates two objects (in String pool and in heap) and one reference variable where the variable 's' will refer to the object in the heap.

Java String Methods

- **Java String IsEmpty()**
  - This method checks whether the String contains anything or not. If the java String is Empty, it returns true else false.
- **Java String toLowerCase()**
  - The java string toLowerCase() method converts all the characters of the String to lower case.
- **Java String toUpper()**
  - The Java String toUpperCase() method converts all the characters of the String to upper case.
- **Java String equals()**
  - The Java String equals() method compares the two given strings on the basis of content of the string i.e Java String representation. If all the characters are matched, it returns true else it will return false.

Interface in Java: Abstraction

- In Java, Data Abstraction is defined as the process of reducing the object to its essence so that only the necessary characteristics are exposed to the users.
- It basically deals with hiding the details and showing the essential things to the user.
- Whenever we get a call, we get an option to either pick it up or just reject it. But in reality, there is a lot of code that runs in the background.

```
abstract class Mobile          // abstract
class mobile
{
    abstract void run();    // abstract
method
}
```

Interface

- An interface is a blueprint of a class or reference type in Java. It is similar to class. It is a collection of abstract methods.
- Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements.
- In an interface, each method is public and abstract but it does not contain any constructor

Defining an Interface

- An interface is defined much like a class. This is a simplified general form of an interface:

```
access-specifier interface interfacename
{
        return-type method-name1(parameter-list);
        return-type method-name2(parameter-list);
        type final-varname1 = value;
        type final-varname2 = value;
        //...
        return-type method-nameN(parameter-list);
        type final-varnameN = value;
}
```

Class

- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
- A class in Java can contain:
  - fields
  - methods
  - constructors
  - blocks
  - nested class and interface

Rules

- A class can have only public or default(no modifier) access specifier.

- It can be either abstract, final or concrete (normal class).
- It must have the class keyword, and class must be followed by a legal identifier.

- It may optionally extend one parent class. By default, it will extend java.lang.Object.

Declaring Classes / Syntax

```
class    MyClass
{
   // Instance Variables
   // method declarations
   //constructor
}
```

```
class Box
{
  // Instance Variables
  float width;
  float height;
  float depth;

  // method declarations
  void getValue(int w, int l, int h);
  void Area(int w, int l, int h);

// Constructor Declaration of Class
  public Box(int w, int l, int h);
 }
```

new Operator in Java

- When we are declaring a class in java, we are just creating a new data type.
- A class provides the blueprint for objects. We can create an object from a class.
- However obtaining objects of a class is a two-step process :
  - Declaration
  - Instantiation and Initialization

Constructor

- A Constructor in java is a block of code similar to a method that initializes the newly created object.

- A constructor resembles an instance method in java but it's not a method as it doesn't have a return type.

- There are two types of constructors in java:
    - Default constructor (No-argument constructor)
    - Parameterized constructor

Dot Operator

```
Box mybox = new Box();
mybox.width=100;
mybox.height=200;
mybox.depth=300;
mybox.getvalue();
mybox.area();
```

Garbage collection

- Garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.

- As in use object, or a referenced object, means that some part of our program still maintains a pointer to that object.

- An unused object, or unreferenced object, is no longer referenced by any part of our program. So the memory used by an unreferenced object can be reclaimed.

- In a programming language like C, C++, etc. allocating and deallocating memory is a manual process. In Java, process of deallocating memory is handled automatically by the garbage collector.

Exception Handling in Java

- Errors arise unexpectedly and can result in disrupting the normal flow of execution.
- This is something that every programmer faces at one point or the other while coding.
- **Java,** being the most prominent **object-oriented language,** provides a powerful mechanism to handle these errors/exceptions.
- An exception is a problem that arises during the execution of a program. It can occur for various reasons say
    - A user has entered an invalid data
    - File not found
    - A network connection has been lost in the middle of communications
    - The JVM has run out of a memory

Exception handling

- An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

- When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred.

- Creating an exception object and handing it to the runtime system is called throwing an exception.

Exception handling

- The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler.
- The search begins with the method in which the error occurred and proceeds through the call stack in the reverse order in which the methods were called.
- When an appropriate handler is found, the runtime system passes the exception to the handler. An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.

Figure 7-9

Java program to demonstrate how exception is thrown

```
1.  class ThrowsExecption
2. {
3.          public static void main(String args[])
4.   {
5.                      String str = null;
6.                      System.out.println(str.length());
7.          }
8. }
```

Output

```
Exception in thread "main" java.lang.NullPointerException
                                        at
ThrowsExecption.main(ThrowsExecption.java:6)
```

Built-in Exceptions

Built-in Exceptions	Description
ArithmeticException	It is thrown when an exceptional condition has occurred in an arithmetic operation.
ArrayIndexOutOfBoundsException	It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
ClassNotFoundException	This exception is raised when we try to access a class whose definition is not found.
FileNotFoundException	An exception that is raised when a file is not accessible or does not open.
IOException	It is thrown when an input-output operation is failed or interrupted.
InterruptedException	It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
NoSuchFieldException	It is thrown when a class does not contain the field (or variable) specified.

User-Defined Exceptions

- Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, a user can also create exceptions which are called ‘User-Defined Exceptions’.

- **Key points to note:**

  - A user-defined exception must extend Exception class.
  - The exception is thrown using *throw* keyword.

Exception Handling in Java

The **Exception Handling in Java** is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Multitasking vs. Multithreading vs. Multiprocessing vs. parallel processing

Multitasking	Multithreading
Ability to execute more than one task at the same time is known as multitasking.	It is a process of executing multiple threads simultaneously. Multithreading is also known as Thread-based Multitasking.
Multiprocessing	**Parallel Processing**
It is same as multitasking, however in multiprocessing more than one CPUs are involved. On the other hand one CPU is involved in multitasking.	It refers to the utilization of multiple CPUs in a single computer system.

Figure 7-15

Thread and Multithreading

- A thread is actually a lightweight process.
- Unlike many other computer languages, Java provides built-in support for multithreaded programming.
- A multithreaded program contains two or more parts that can run **concurrently**.
- Each part of such a program is called thread and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking.
- Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.
- Each part of such program is called a thread. So, threads are light-weight sub-processes within a process

Creating a user-defined thread in Java

- There are two ways to create a thread in Java:
  - By extending Thread class.
  - By implementing Runnable interface

Extending Java Thread

- The first way to create a thread is to create a new class that extends Thread, then override the run() method and then to create an instance of that class.
- The run() method is what is executed by the thread after we call start().

```
 public class MyClass extends Thread
{
        public void run()
        {
                System.out.println("MyClass running");
        }
}
```

Thread creation by extending Thread class

```
class MultithreadingDemo extends Thread
{
  public void run()
  {
    System.out.println("My thread is in running state.");
  }
  public static void main(String args[])
  {
     MultithreadingDemo obj=new MultithreadingDemo();
     obj.start();
  }
}
```

Runnable Interface

- The easiest and second way to create a thread is to create a class that implements the Runnable interface.
- To implement Runnable interface, a class need only implement a single method called run( ), which is declared like this:
  - **public void run( ) {}**
- Inside run( ), we will define the code that constitutes the new thread.

```
public class MyClass implements Runnable
{
      public void run()
      {
            System.out.println("MyClass
running");
      }
}
```

Java Applet

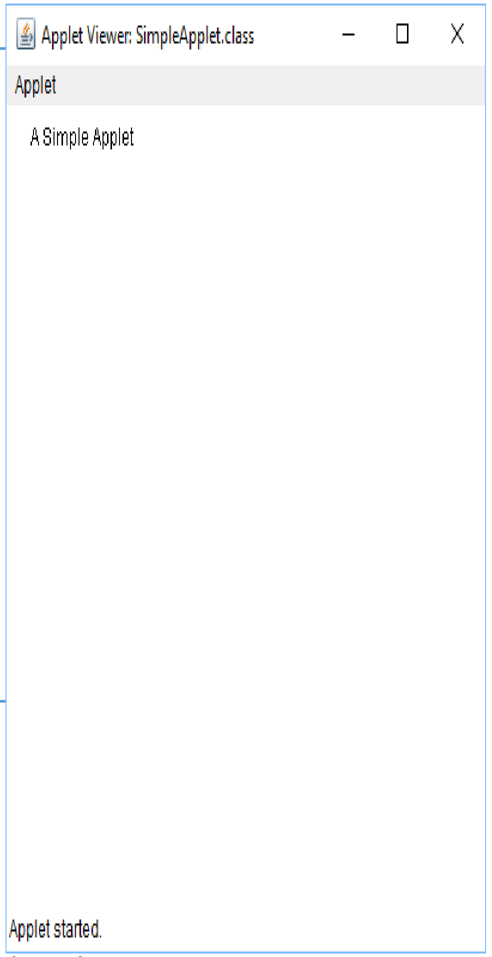
- A Java Applet is a small software program which can be transferred over HTTP.
- Applets in Java are small and dynamic internet-based programs.
- A Java Applet can be only executed within the applet framework of Java. For an easy execution of applets, a restricted ‘sandbox’ is provided by the applet framework.
- Generally, the applet code is embedded within an HTML page.
- The applet codes are executed when the HTML pages get loaded into the Java-compatible web browsers.

Applet Hierarchy in Java

- class java.lang.**Object**
  - class java.awt.**Component**
    - class java.awt.**Container**
      - class java.awt.**Panel**
        - » class java.applet.**Applet**
- The Java Applet class which is a class of applet package extends the Panel class of awt package. The Panel class is a subclass of the Container class of the same package. The Container class is an extension of Component class belonging to the same package. The Component class is an abstract class and derives several useful classes for the components such as Checkbox, List, buttons, etc.

Java Applet - Example

```
import java.awt.*;
import java.applet.*;

public class SimpleApplet extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("A Simple
Applet", 20, 20);
        }
}
```



A Simple Applet

Applet started.

Applet Life Cycle

- **public void init()**
  - This is the very first method to be invoked during the life cycle of an applet. In this method, the variable that will be used further in the applet is initialized. One thing we must note here is that this method can be invoked only once per applet life cycle.
- **public void start()**
  - This is the second method that is invoked just after the init() method is called by the browser. Each time a user revisits the web page containing the applet, start() method is invoked and the applet is started.

Applet Life Cycle

- **public void stop()**
  - This method is invoked whenever a user leaves the web page containing applet. In other words, the stop() method is used to suspend the threads which are not required when the applet is in the background or is not visible on the screen. These can be easily resumed using the start() method.
- **public void destroy()**
- Finally, we have the destroy() method which is invoked in order to completely remove an applet from the memory. This method is invoked only once per applet life cycle and all the engaged resources must be freed up before this method is called. **public void paint(Graphics g)**
  - This method is invoked whenever an applet needs to be redrawn or repainted in the browser, irrespective of the cause. The paint() method takes one Graphic object as a parameter that contains the graphics context in which the applet is being executed. Also, this method is invoked each time output is expected from the applet

Step for applet development

1. Edit a Java source code and related HTML file.

2. Compile our java program.

3. Execute the **appletviewer**, specifying the name of our applet's source file. The appletviewer will encounter the APPLET tag within the comment and execute our applet.

How to run an Applet Program

To compile an applet	
	C:\> javac appletfilename.java
To run the applet	
	C:\>appletviewer htmlfile.html

Java Package

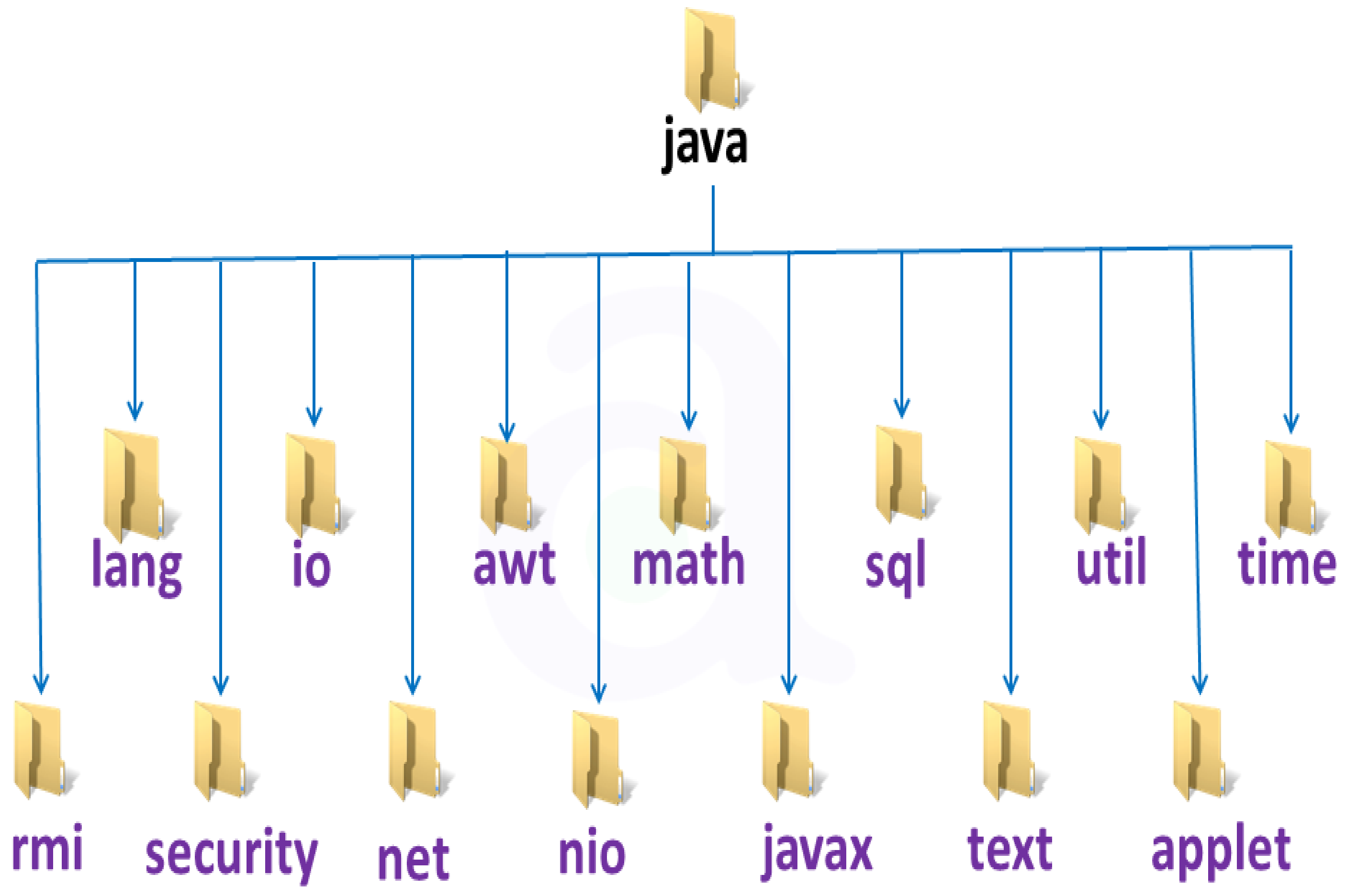
- Package is a mechanism to encapsulate a group of classes, sub packages and interfaces.

- Packages are used for:
  - Preventing naming conflicts.

    - For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee

  - Making searching/locating and usage of classes, interfaces, enumerations and annotations easier.

Java Package



Creating Package

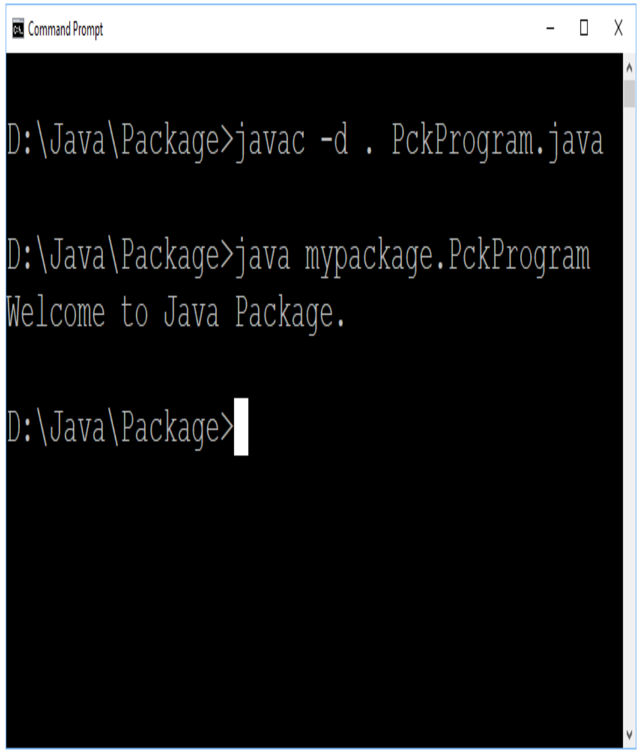
- Creating a package in java is quite easy. Simply include a package command followed by name of the package as the first statement in java source file.

```
package pkg;
```

```
package
mypackage;
public        class
Student
{
  ...statement;
}
```

Creating Package

```
package mypackage;
public class PckProgram
{
        public static void
main(String args[])
        {

System.out.println("Welcome
to Java Package.");
        }
}
```



```
D:\Java\Package>javac -d . PckProgram.java

D:\Java\Package>java mypackage.PckProgram
Welcome to Java Package.

D:\Java\Package>
```

Java Package

How to compile java package

Syntax:	`javac -d directory javafilename`
Example:	`javac -d . PckProgram.java`

How to run java package program?

To Compile:	**How to run java package program?**
To Run:	How to run java package program?

Abstract window toolkit

- Abstract Window Toolkit acronym as AWT is a toolkit of classes in Java which helps a programmer to develop Graphics and Graphical User Interface components.
- It is a part of JFC (Java Foundation Classes) developed by Sun Microsystems.
- The AWT API in Java primarily consists of a comprehensive set of classes and methods that are required for creating and managing the Graphical User Interface(GUI) in a simplified manner.
- It was developed for providing a common set of tools for designing the cross-platform GUIs.

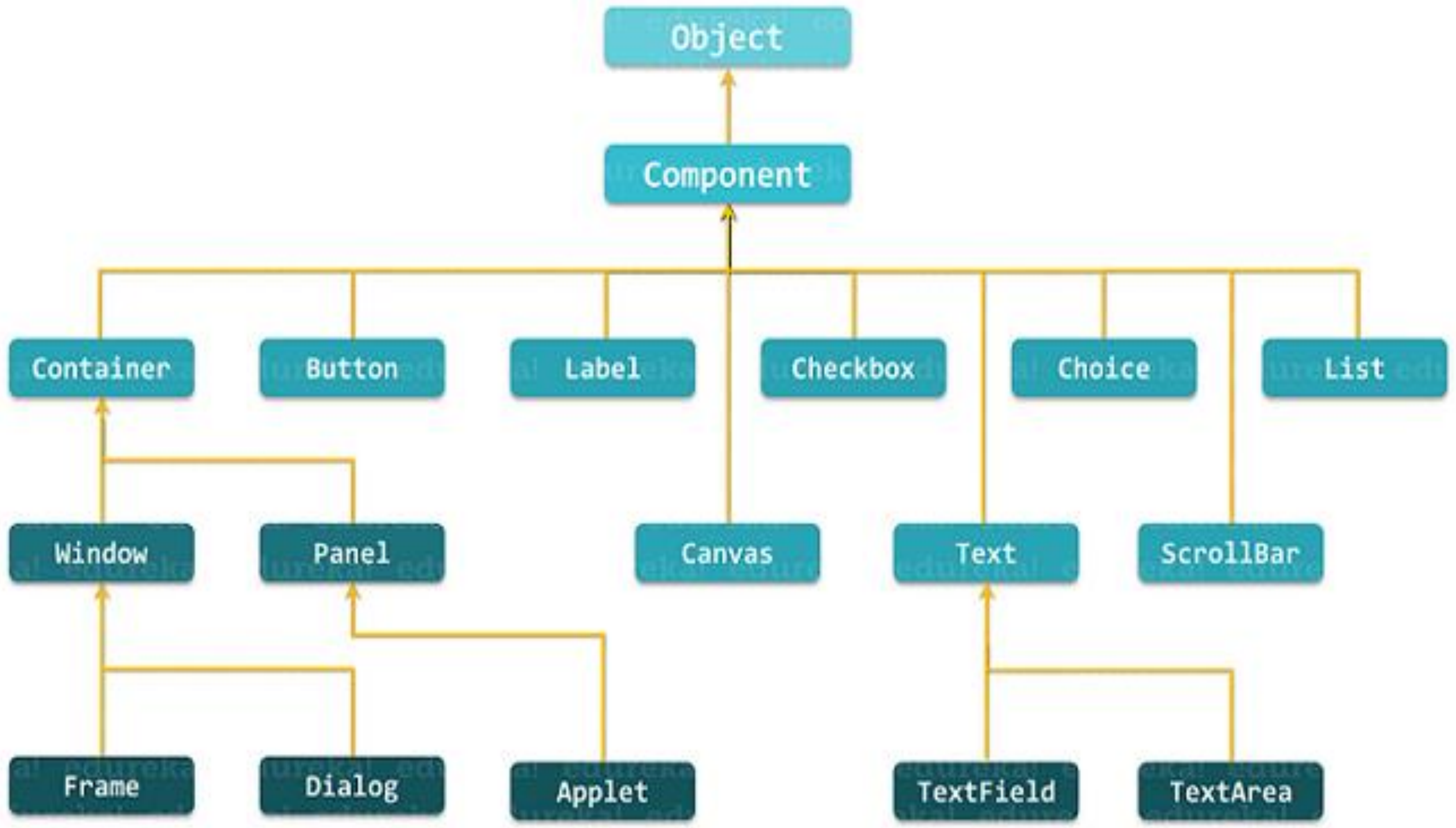
Abstract window toolkit

- One of the important features of AWT is that it is platform dependent. This means that the AWT tools use the native toolkits of the platforms they are being implemented.

- This approach helps in preserving the look and feel of each platform. But as said everything comes with a price, there is a major drawback of this approach.

- When executed on various platforms because of platform dependency it will look different on each platform. This hampers the consistency and aesthetics of an application.

Hierarchy Of AWT



AWT Controls in Java

- The AWT supports the following types of controls:
  - Labels
  - Push buttons
  - Check boxes
  - Choice lists
  - Lists
  - Scroll bars
  - Text Area
  - Text Field
- These controls are subclasses of Component.

Layout

- Layout means the arrangement of components within the container.
- In other way we can say that placing the components at a particular position within the container.
- The task of layouting the controls is done automatically by the Layout Manager.

AWT and Swing classes

Several AWT and Swing classes provide layout managers for general use:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout