Primitive DataTypes

Data Type	Size	Description
`Byte`	1 byte	Stores whole numbers from -128 to 127
`Short`	2 bytes	Stores whole numbers from -32,768 to 32,767
`Int`	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
`Long`	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`Float`	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
`Double`	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
`Boolean`	1 bit	Stores true or false values
`Char`	2 bytes	Stores a single character/letter or ASCII values

ASCII - Binary Character Table

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
A	097	01100001	A	065	01000001
B	098	01100010	B	066	01000010
C	099	01100011	C	067	01000011
D	100	01100100	D	068	01000100
E	101	01100101	E	069	01000101
F	102	01100110	F	070	01000110
G	103	01100111	G	071	01000111
H	104	01101000	H	072	01001000
I	105	01101001	I	073	01001001
J	106	01101010	J	074	01001010
K	107	01101011	K	075	01001011
L	108	01101100	L	076	01001100
M	109	01101101	M	077	01001101
N	110	01101110	N	078	01001110
O	111	01101111	O	079	01001111
P	112	01110000	P	080	01010000
Q	113	01110001	Q	081	01010001
R	114	01110010	R	082	01010010
S	115	01110011	S	083	01010011
T	116	01110100	T	084	01010100
U	117	01110101	U	085	01010101
V	118	01110110	V	086	01010110
W	119	01110111	W	087	01010111
X	120	01111000	X	088	01011000
Y	121	01111001	Y	089	01011001
Z	122	01111010	Z	090	01011010

For loop	While loop
Initialization may be either in loop statement or outside the loop.	Initialization is always outside the loop.
Once the statement(s) is executed then after increment is done.	Increment can be done before or after the execution of the statement(s).
It is normally used when the number of iterations is known.	It is normally used when the number of iterations is unknown.
Condition is a relational expression.	Condition may be expression or non-zero value.
It is used when initialization and increment is simple.	It is used for complex initialization.
For is entry controlled loop.	While is also entry controlled loop.
for (init ; condition ; iteration) { statement(s); }	while (condition) { statement(s); }

Difference Between Method Nd Constructor

1	Purpose	Constructor is used to create and initialize an Object .	Method is used to execute certain statements.
2	Invocation	A constructor is invoked implicitly by the System.	A method is to be invoked during program code.
3	Invocation	A constructor is invoked when new keyword is used to create an object.	A method is invoked when it is called.
4	Return type	A constructor can not have any return type.	A method can have a return type.
5	Object	A constructor initializes an object which is not existent.	A method can be invoked only on existing object.
6	Name	A constructor must have same name as that of the class.	A method name can not be same as class name.
7	Inheritance	A constructor cannot be inherited by a subclass.	A method is inherited by a subclass.

Overriding:Same method name and different function are there.

Overloading:passing

Difference Between Method Overloading Method Overriding

Method Overloading	Method Overriding
Method overloading is a compile-time polymorphism.	Method overriding is a run-time polymorphism.
It helps to increase the readability of the program.	It is used to grant the specific implementation of the method which is already provided by its parent class or superclass.
It occurs within the class.	It is performed in two classes with inheritance relationships.
Method overloading may or may not require inheritance.	Method overriding always needs inheritance.
In method overloading, methods must have the same name and different signatures.	In method overriding, methods must have the same name and same signature.
In method overloading, the return type can or can not be the same, but we just have to change the parameter.	In method overriding, the return type must be the same or co-variant.
Static binding is being used for overloaded methods.	Dynamic binding is being used for overriding methods.
Poor Performance due to compile time polymorphism.	It gives better performance. The reason behind this is that the binding of overridden methods is being done at runtime.
Private and final methods can be overloaded.	Private and final methods can’t be overridden.
Argument list should be different while doing method overloading.	Argument list should be same in method overriding.
Early Binding	Late Binding

Scanner

How to get input from user in Java

Java Scanner Class

Java **Scanner class** allows the user to take input from the console. It belongs to **java.util** package. It is used to read the input of primitive types like int, double, long, short, float, and byte. It is the easiest way to read input in Java program.

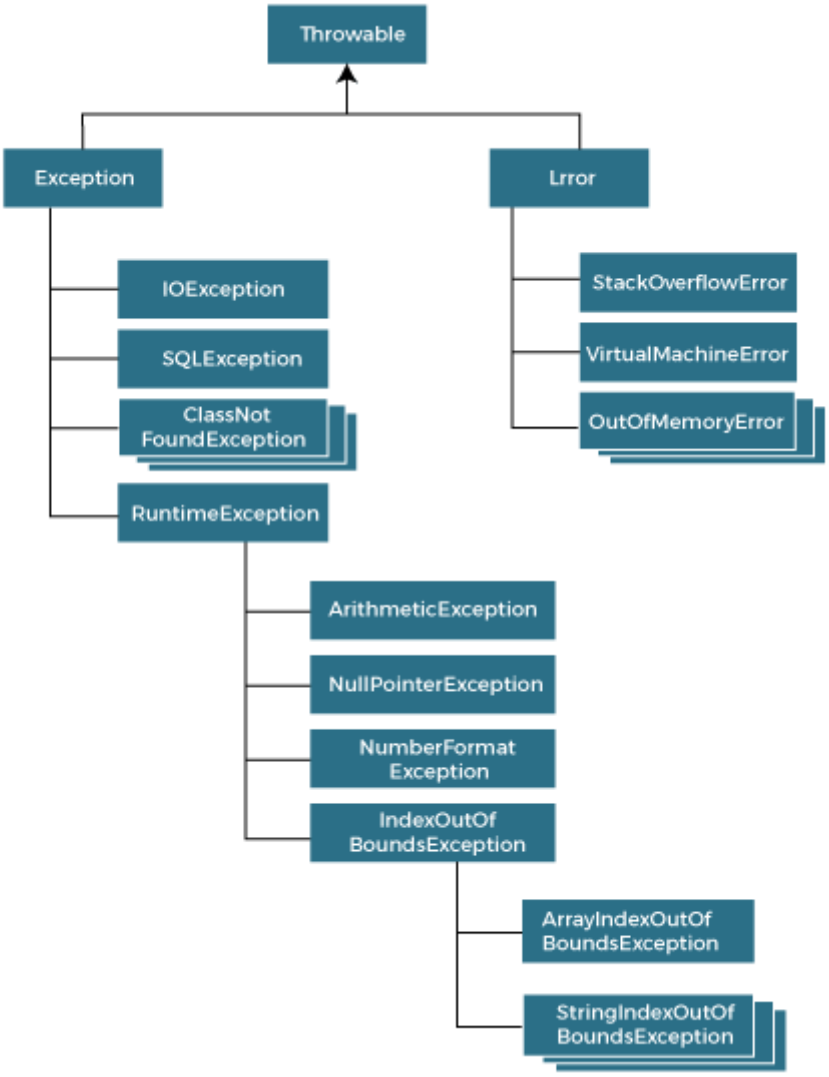
Syntax

1. Scanner sc=**new** Scanner(System.in);

Method	Description
int nextInt()	It is used to scan the next token of the input as an integer.
float nextFloat()	It is used to scan the next token of the input as a float.
double nextDouble()	It is used to scan the next token of the input as a double.
byte nextByte()	It is used to scan the next token of the input as a byte.
String nextLine()	Advances this scanner past the current line.
boolean nextBoolean()	It is used to scan the next token of the input into a boolean value.
long nextLong()	It is used to scan the next token of the input as a long.
short nextShort()	It is used to scan the next token of the input as a Short.
BigInteger nextBigInteger()	It is used to scan the next token of the input as a BigInteger.
BigDecimal nextBigDecimal()	It is used to scan the next token of the input as a BigDecimal.

Hierarchy of Java Exception classes

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1. Checked Exception

2. Unchecked Exception
3. Error

Difference between Checked and Unchecked Exceptions

1) Checked Exception

The classes that directly inherit the Throwable class except Runtime Exception and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes that inherit the Runtime Exception are known as unchecked exceptions. For example,

ArithmeticException,NullPointerException,ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3) Error

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

In Java, 3 streams are created for us automatically. All these streams are attached with the console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

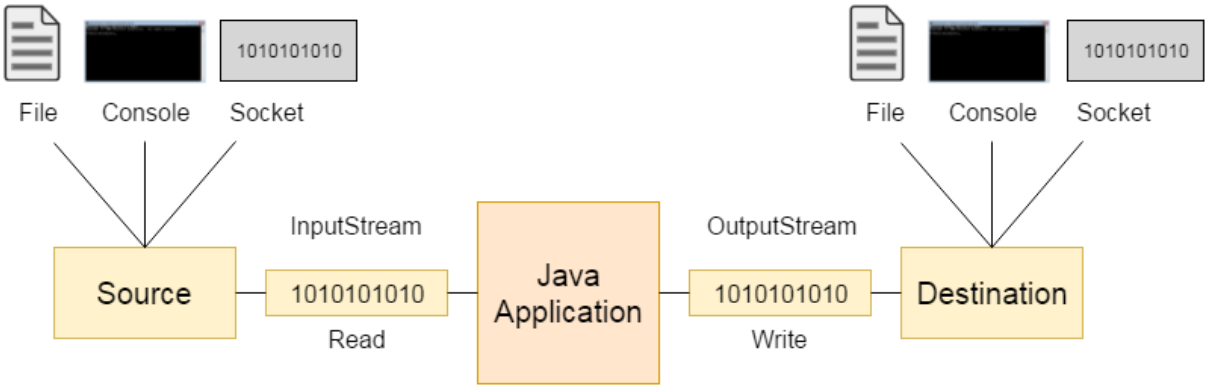
OutputStream

Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.

InputStream

Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

Let's understand the working of Java OutputStream and InputStream by the figure given below.



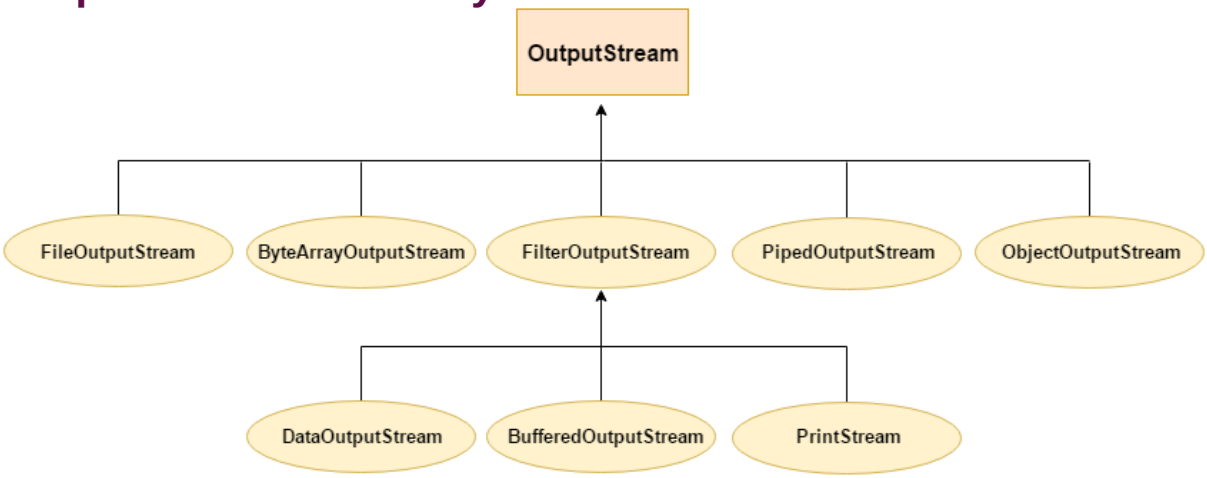
OutputStream class

OutputStream class is an abstract class. It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Useful methods of OutputStream

Method	Description
1) public void write(int)throws IOException	is used to write a byte to the current output strean
2) public void write(byte[])throws IOException	is used to write an array of byte to the current out
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

OutputStream Hierarchy



InputStream class

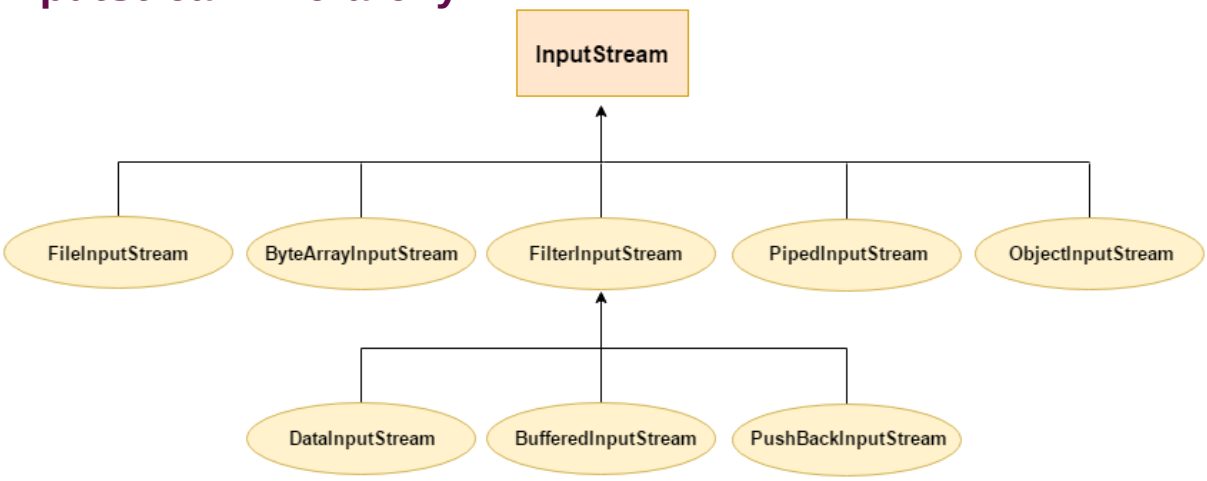
InputStream class is an abstract class. It is the superclass of all classes representing an input stream of bytes.

Useful methods of InputStream

Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read stream.

3) public void close()throws IOException	is used to close the current input stream.

InputStream Hierarchy



Difference Between String Buffer and String Builder

StringBuffer Class	StringBuilder Class
StringBuffer is present in Java.	StringBuilder was introduced in Java 5.
StringBuffer is synchronized. This means that multiple threads cannot call the methods of StringBuffer simultaneously.	StringBuilder is asynchronized. This means that multiple threads can call the methods of StringBuilder simultaneously.
Due to synchronization, StringBuffer is called a thread safe class.	Due to its asynchronous nature, StringBuilder is not a thread safe class.
Due to synchronization, StringBuffer is lot slower than StringBuilder.	Since there is no preliminary check for multiple threads, StringBuilder is a lot faster than StringBuffer.
It is Thread Safe.	It is Not Thread safe.

Class Declaration of StringBuilder

The java.lang.StringBuilder class is a part of java.lang package and has the following class declaration:

public final class StringBuilder

extends Object

implements Serializable, CharSequence

Looking at the Constructors of StringBuilder in Java

The following table lists and describes the constructors of StringBuilder in Java

Constructor Name	Description
StringBuilder()	It constructs a blank string builder with a capacity of 16 characters
StringBuilder(int capacity)	It creates an empty string builder with the specified capacity

StringBuilder(CharSequence seq)	It creates a string builder with the same characters specified as the argument
StringBuilder(String str)	It will construct a string builder with the string specified in the argument

Method	Description
StringBuilder append (String s)	This method appends the mentioned string with the existing string. You can also with arguments like boolean, char, int, double, float, etc.
StringBuilder insert (int offset, String s)	It will insert the mentioned string to the other string from the specified offset position. Like append, you can overload this method with arguments like (int, boolean), (int, int), (int, char), (int, double), (int, float), etc.
StringBuilder replace(int start, int end, String s)	It will replace the original string with the specified string from the start index till the end index.
StringBuilder delete(int start, int end)	This method will delete the string from the mentioned start index till the end index.
StringBuilder reverse()	It will reverse the string.
int capacity()	This will show the current StringBuilder capacity.

void ensureCapacity(int min)	This method ensures that the StringBuilder capacity is at least equal to the mentioned minimum.
char charAt(int index)	It will return the character at the specified index.
int length()	This method is used to return the length (total characters) of the string.
String substring(int start)	Starting from the specified index till the end, this method will return the substring.
String substring(int start, int end)	It will return the substring from the start index till the end index.
int indexOf(String str)	This method will return the index where the first instance of the specified string occurs.
int lastIndexOf(String str)	It will return the index where the specified string occurs the last.

Void trimToSize()	It will attempt to reduce the size of the StringBuilder.

StringBuffer

Methods	Action Performed
append()	Used to add text at the end of the existing text.
length()	The length of a StringBuffer can be found by the length( ) method
capacity()	the total allocated capacity can be found by the capacity( ) method
charAt()	This method returns the char value in this sequence at the specified index.
delete()	Deletes a sequence of characters from the invoking object
deleteCharAt()	Deletes the character at the index specified by *loc*
ensureCapacity()	Ensures capacity is at least equals to the given minimum.
insert()	Inserts text at the specified index position
length()	Returns length of the string
reverse()	Reverse the characters within a StringBuffer object
replace()	Replace one set of characters with another set inside a StringBuffer object