Java

Module 4

Lang Package

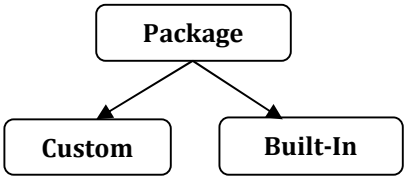
Author
Srinivas Dande

My JAVA LEARNING CENTER

4.1 Packages

- **Packages are used to group the classes, interfaces, Enums and Annotations which have similar functionalities.**

Types of Package



Custom Packages

- **The packages which are developed by the developer as per application or project requirements are called as Custom package or User Defined package.**

Built-In Package

- **The packages which are already developed and provided along with Java and other technologies are called as Built-In package or Pre-Defined Package.**
- **You can just import built-in packages and use various functionalities directly.**
- **Following are the commonly used built-in packages:**
  - **java.lang**
  - **java.lang.reflect**
  - **java.util**
  - **java.io**
  - **java.sql**
  - **java.math**
  - **java.awt**
  - **java.nio (From Java 7)**
  - **javax.sql**

    etc.
- **An application programming interface (API) is a collection of predefined packages, classes, and interfaces with their methods, fields and constructors.**
- **All the Java API packages are prefixed with java or javax.**
- **Core packages of java starts with java.**
- **Advanced packages of java starts with javax.**

4.2 java.lang Package

- java.lang package contains the classes and interfaces which are required for basic Java language functionalities.
- java.lang package is by default imported in all the java programs i.e you can access all the classes and interfaces of java.lang package in your Java programs without importing.

Commonly used classes from java.lang package			
Object	String	StringBuffer	StringBuilder (From Java 5)
Boolean	Character	Number	Byte
Short	Integer	Long	Float
Double	Void	Math	Process
System	Runtime	Throwable	Exception
Error	Thread	ThreadGroup	ClassLoader
Class	etc		

Commonly used interfaces from java.lang package			
Cloneable	Comparable	CharSequence	AutoCloseable (From Java 7)
Runnable	etc		

4.2.1 Object class

- Object is predefined class in java.lang package.
- Object is default super class for all the Java classes i.e. you can refer all the members of Object class in your program without extending.
- When one class is defined without any extends keyword then Object class will be the direct super class otherwise Object class will be indirect super class.
- Object class doesn't have any super class.
- The members available in this class can be referred with any type of reference like class or interface or array etc.
- Object class reference variable can hold any type of object.

Members of java.lang.Object class

```
public class Object{
        public Object();
        public final native Class getClass();
        public native int hashCode();
        public String toString();
        public boolean equals(Object);
        protected native Object clone(); throws CloneNotSupportedException
        protected void finalize(); throws Throwable
        public final native void notify();
        public final native void notifyAll();
        public final void wait(); throws InterruptedException
        public final native void wait(long);       throws InterruptedException
        public final void wait(long, int);  throws InterruptedException
  }
```

Lab511.java

```
class Hello{
Hello(int a){
super(a);
}
}

class Lab511{
public static void main(String as[]){
Hello h= new Hello(99);
}
}
```

4.2.2 Exploring Methods of Object class

public final native Class getClass()

- When JVM loads the class in main memory then it will create one default object of the type java.lang.Class.
- You can access complete information about your class by using that default object of type java.lang.Class.
- getClass() method can be used to access the default object created by JVM at the time of loading.

Lab512.java

```
class Hello{
void show(){
System.out.println("Hello - show");
}
}

class Lab512{
public static void main(String as[]){
Hello h= new Hello();
h.show();
Class myclass = h.getClass();
System.out.println(myclass.getName());
System.out.println(myclass.getSuperclass());
System.out.println(myclass.getPackage());
}
}
```

Lab513.java

```
package com.myjlc;

class Hai{ }
class Hello extends Hai {
void show(){
System.out.println("Hello - show");
}
}

class Lab513{
public static void main(String as[]){
Hello h= new Hello();
h.show();
Class myclass = h.getClass();
System.out.println(myclass.getName());
System.out.println(myclass.getSuperclass());
System.out.println(myclass.getPackage());
}
}
```

public native int hashCode()

- When you create an object then JVM assigns some integer value for that object. This integer value is called as Hash Code value of the object.
- hashCode() method can be used to access Hash Code value of an object.
- The Hash Code value will be used internally when we manage the collection of object using some Built Collection API.
- You can override hashCode() method in your class to provide your own implementation.
- Usually hashCode() method is overridden in many java built-in classes to generate hashCode using contents of objects.

Lab514.java

```
class Hello  {
int a;
Hello(int a){
this.a=a;
}
void show(){
System.out.println("show() - a => "+a);
}
}

class Lab514{
public static void main(String as[]){
Hello hello1= new Hello(100);
hello1.show();
Hello hello2= new Hello(200);
hello2.show();

int hash1 = hello1.hashCode();
System.out.println(hash1 );
int hash2 = hello2.hashCode();
System.out.println(hash2 );

}
}
```

Lab515.java

```
class Hello  {
int a;
Hello(int a){
this.a=a;
}
void show(){
System.out.println("show() - a => "+a);
}
public int hashCode(){
return a*25;
}
}

class Lab515{
public static void main(String as[]){
Hello hello1= new Hello(100);
hello1.show();
Hello hello2= new Hello(200);
hello2.show();

int hash1 = hello1.hashCode();
System.out.println(hash1 );
int hash2 = hello2.hashCode();
System.out.println(hash2 );
}
}
```

public String toString()

- When you print reference variable then following tasks will happen:
  - If reference variable contains null then null value will be displayed.
  - If reference variable contains address of an object then toString() method will be invoked by the JVM automatically.
  - By default toString() of object class will print:

    <fullyQualifiedClassName>@<HexadecimalOfHashCode>
- You can override this method in your class to display some meaningful String.
- toString() method returns the String representation of an object.Usually this method is used to print contents of an object. This method is already overridden in many java built-in classes like String, StringBuffer, Integer etc.

Lab516.java

```
class Hello  {
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
}

class Lab516{
public static void main(String as[]){
Hello h1 = null;
System.out.println(h1);

Hello h2 = new Hello(10,20);
System.out.println(h2);
System.out.println(h2.toString());
System.out.println(h2.hashCode());
}
}
```

Lab517.java

```
class Hello  {
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
public int hashCode(){
return a+b;
}
}

class Lab517{
public static void main(String as[]){
Hello h1 = null;
System.out.println(h1);

Hello h2 = new Hello(10,20);
System.out.println(h2);
System.out.println(h2.toString());
System.out.println(h2.hashCode());
}
}
```

Lab518.java

```
class Hello  {
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
public String toString(){
String str = "[a = "+a+", b = "+b+"]";
return str;
}
}
```

```
class Lab518{
public static void main(String as[]){
Hello h1 = null;
System.out.println(h1);

Hello h2 = new Hello(10,20);
System.out.println(h2);
System.out.println(h2.toString());
}
}
```

Lab519.java

```
class Customer{

int cid;
String cname;
String email;
long phone;

Customer(){}

Customer(int cid,String cname,String email,long
phone){
this.cid=cid;
this.cname=cname;
this.email=email;
this.phone=phone;
}
public String toString(){
String
str="["+cid+","+cname+","+email+","+phone+"]";
return str;
}
}

class Lab519{
public static void main(String as[]){
Customer cust1 = new Customer();
System.out.println(cust1);

Customer cust2 = new
Customer(101,"Srinivas","sri@mjlc.com",12345);
System.out.println(cust2);
}
}
```

public boolean equals(Object obj)

- You can use == operator to compare two variables of primitive type or reference type.
- When you compare primitive type variables using == operator then it will compare the values available in primitive variables.
- When you compare reference type variables using == operator then it will compare the addresses available in reference variables.
- When you want to compare the contents of two objects then you can use equals() method.
- You can use following versions of equals() method :
    - Object class equals() method.
    - Overridden equals() method.
- Object class equals() method will always compares the address of two objects.

```
public boolean equals (Object obj){
  return this == obj;
 }
```

- You can override equals() method in your class when you want to compare contents of two objects.
- Usually equals() method is overridden in many java built-in classes to compare contents of objects.

Lab520.java

```
class Hello{
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
}
class Lab520{
public static void main(String as[]){
Hello h1=new Hello(10,20);
Hello h2=new Hello(10,20);
Hello h3=new Hello(50,60);
Hello h4=h3;

System.out.println(h1==h2);
System.out.println(h1==h3);
System.out.println(h2==h3);
System.out.println(h3==h4);
}
}
```

Lab521.java

```
class Hello{
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
}
class Lab521{
public static void main(String as[]){
Hello h1=new Hello(10,20);
Hello h2=new Hello(10,20);
Hello h3=new Hello(50,60);
Hello h4=h3;

System.out.println(h1.equals(h2));
System.out.println(h1.equals(h3));
System.out.println(h2.equals(h3));
System.out.println(h3.equals(h4));
}
}
```

Lab522.java

```
class Hello{
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
//Implementation Same as Object class
public boolean equals(Object obj){
return (this==obj);
}
}
class Lab522{
public static void main(String as[]){
Hello h1=new Hello(10,20);
Hello h2=new Hello(10,20);
Hello h3=new Hello(50,60);
Hello h4=h3;

System.out.println(h1.equals(h2));
System.out.println(h1.equals(h3));
System.out.println(h2.equals(h3));
System.out.println(h3.equals(h4));
}
}
```

Lab523.java

```
class Hello{
int a;
int b;
Hello(int a,int b){
this.a=a;
this.b=b;
}
public boolean equals(Object obj){
Hello h=(Hello)obj;
return this.a == h.a && this.b == h.b;
}
}
class Lab523{
public static void main(String as[]){
Hello h1=new Hello(10,20);
Hello h2=new Hello(10,20);
Hello h3=new Hello(50,60);
Hello h4=h3;

System.out.println(h1.equals(h2));
System.out.println(h1.equals(h3));
System.out.println(h2.equals(h3));
System.out.println(h3.equals(h4));
}
}
```

Lab524.java

```
class Customer{

int cid;
String cname;
String email;
long phone;

Customer(){
}

Customer(int cid,String cname,String email,long
phone){
this.cid=cid;
this.cname=cname;
this.email=email;
this.phone=phone;
}
public boolean equals(Customer cust){
return this.cid == cust.cid &&
this.cname.equals(cust.cname) &&
this.email.equals(cust.email) && this.phone ==
cust.phone;
}
}

class Lab524{
public static void main(String as[]){
Customer cust1 = new
Customer(101,"Srinivas","sri@mjlc.com",12345);
Customer cust2 = new
Customer(101,"Srinivas","sri@mjlc.com",12345);
Customer cust3 = new
Customer(102,"Srinivas","vas@mjlc.com",67890);

System.out.println(cust1.equals(cust2));
System.out.println(cust1.equals(cust3));
}
}
```

4.2.3 Memory Management

- When you are starting the JVM then JVM will request some memory from the OS. If OS allocates the required memory then JVM will start otherwise the error message will be displayed and JVM will not start.

Lab525.java

```
class Lab525{
public static void main(String as[]){
// Runtime rt = new Runtime();

Runtime rt =Runtime.getRuntime();

System.out.println(rt.maxMemory());
System.out.println(rt.totalMemory());
System.out.println(rt.freeMemory());
}
}
```

Execute as follows

```
java Lab525
java -Xms512m Lab525
java -Xms1024m Lab525

java -Xmx2048m Lab525
java -Xmx5120m Lab525

java -Xmx5120m -Xms1024m Lab525

java -Xms20480m Lab525

        -Xms    => Initial Memory
        -Xmx    => Max Memory
```

- When a JVM executes a program, it needs memory to store many things like:
  - Byte codes and other information extracted from loaded class files.
  - Objects created in your program.
  - Java Method Calls
  - Native Method Calls
  - Methods Parameters.
  - Values returned from methods.
  - Local variables declared.
  - Intermediate results of computations.

- The Java virtual machine organizes the memory into several runtime data areas. Some are
  - Method Area / Class Heap
  - Heap Memory
  - Stack Memory

Method Area

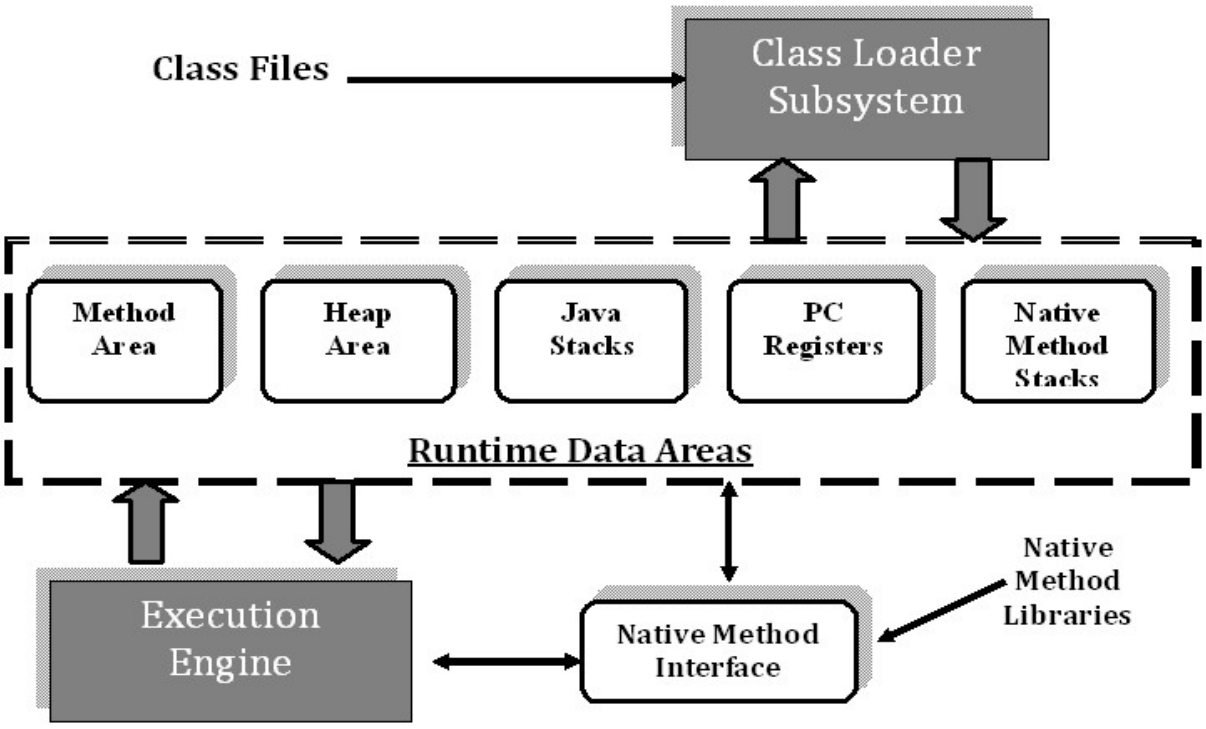
- When the class is loaded by the JVM then the class information will be stored into the method area.
- Method Area is also called as Class Heap.
- Static variable also gets the memory in method area.
- Each instance of the Java virtual machine has one method area.
- This area is shared by all threads running inside the virtual machine.
- These memories should not be de-allocated during the execution of the application.

Heap Memory

- JVM places all objects into the heap memory.
- Each instance of the Java virtual machine has one heap area.
- This area is shared by all threads running inside the virtual machine.
- If you have the reference of the object in your application then it is known as USED or LIVE object.
- If you do not have the reference for the object in your application then the object is known as UNUSED or DEAD object and is called as garbage.
- There is no guarantee that memory for the UNUSED or DEAD objects will be de-allocated immediately.
- When you create the object and required memory is not available in the HEAP then OutOfMemoryError will be thrown by JVM.

Stack Memory

- Stack memory keep track of each and every method invocations. This is called Stack Frame.
- Each thread has its own pc register (program counter) and Java stack frame.
- When the method or constructor is invoked then the implementation will be copied into the stack memory on the top of the Stack. After completing the execution of the method or constructor, memory from the stack will be de-allocated.
- Memory for Local variables and method arguments are allocated in Stack memory.
- The program counter always keeps track of the current instruction which is being executed. After execution of an instruction, the JVM sets the PC to next instruction.
- When you invoke method or constructor and required memory is not available in the STACK then StackOverflowError will be thrown by JVM.



4.2.4 Garbage Collection / finalize() method

- In the case of C, you can use malloc() or calloc() functions to allocate the memory and free() function to de-allocate that memory.
- In the case of C++, you can use new operator to allocate the memory and delete operator to de-allocate the memory which is allocated by new operator.
- In the case of Java, you can use new operator to allocate the memory but there is no operator or function is designed to de-allocate the memory explicitly.
- In Java, automatic memory cleaning process is designed to reduce the developer burden.
- The process of cleaning the memory of unused or un-referred objects by the JVM using GC thread is called as Garbage Collection.
- JVM is responsible to identify the unused objects and de-allocates the memory by invoking GC.
- GC is a service thread which is running inside the JVM.

JVM uses the following techniques to identify the unused objects:

Case1:

When you are accessing any instance members of class dynamically i.e without storing the address in reference variable then after accessing that member's object will be eligible for gc.

```
new Hello().m1();
```

Case 2:

When you are assigning null to reference variable then the object pointed by reference variable will be unused or un-referred. That object is eligible for garbage collection.

```
Hello h=new Hello();
h = null;
```

Case 3:

When you are assigning one reference variable to another reference variable then the object pointed by the reference variable which is left side of assignment operator will be unused or un-referred. That object will be eligible for garbage collection.

```
Hello h1 = new Hello();
Hello h2 = new Hello();
h1 = h2;
```

Case 4:

When object is out of scope i.e. when you are creating object locally either inside methods, blocks or constructors then outside that block object will be eligible for GC.

```
void m1(){
   Hai hai1 = new Hai();
   Hai hai2 = new Hai();
   }
```

- **Sometimes some objects may refuse to clean the memory because those objects still holding some connections like JDBC connections, IO Connections or some network connections.**
- **If you release those resources then GC can clean the memory without any problem.**

- **JVM will invoke runFinalization() method and runFinalization() method will invoke finalize() method on all unused objects.**

- **If you want to release the resources then you can override the finalize() method of Object class inside your class and you can write the resource cleanup code.**

Complete process of Garbage Collection

1) **List of unused objects will be prepared.**
2) **runFinalization() method will be called to release the resources.**
3) **GC will be called to clean the memory of unused objects.**

- **You or JVM can invoke runFinalization() method as follows:**

```
System.runFinalization();
                or
Runtime rt = Runtime.getRuntime();
  rt.runFinalization();
```

- **You or JVM can invoke gc() method as follows:**

```
System.gc();
        or
Runtime rt = Runtime.getRuntime();
  rt.gc();
```

- Object class finalize() method has no implementations.
  - protected void finalize() {}
- So you have to override finalize() method in your class to release the resources used in your class.
- Since GC is a thread so we can request JVM to invoke GC, we can not force to invoke GC immediately.

Lab526.java

```
class Hai{
int a=99;

public void finalize(){
System.out.println("Hai-finalize()");
}
}

class Hello{
Hai hai = new Hai();
int x=10;

void m1(){
System.out.println("Hello-m1()");
Hai hai1 = new Hai();
Hai hai2 = new Hai();
Hai hai3 = new Hai();
}

void show(){
System.out.println("Hello-show()");
}
public void finalize(){
//Resource Cleanup Code
System.out.println("Hello-finalize()");
this.hai = null;
}
}
```

```
class Lab526{
public static void main(String as[]){
new Hello().show();
Hello h=new Hello();
h=null;
Hello h1=new Hello();
Hello h2=new Hello();
h1=h2;
new Hello().m1();
System.runFinalization();
System.gc();
}
}
```

- If the memory space will not be available then JVM will findout the list of unused object. If no unused object will be found then JVM will throw the memory error.
  - java.lang.OutOfMemoryError

SUMMARY

1) Use javap to verify the signature of any class or interface members:

```
javap <fullyQualifiedTypeName>
Ex:
  javap Object                -> INVALID
  javap java.lang.Object      -> VALID
```

2) Following are various implementations of equals() method:

A) Compare sid values

```
public boolean equals(Object obj) {
  boolean flag=false;
  if (obj instanceof Student) {
     Student st = (Student) obj;
     flag = this.sid == st.sid;

  }
  return flag;
}
```

B) Compare sname values

```
public boolean equals(Object obj) {
  boolean flag=false;
  if (obj instanceof Student) {
     Student st = (Student) obj;
     flag = this.sname.equals(st.sname);

  }
  return flag;
}
```

C) Compare sname references

```
public boolean equals(Object obj) {
  boolean flag=false;
  if (obj instanceof Student) {
     Student st = (Student) obj;
     flag = this.sname==st.sname;
  }
  return flag;
}
```

D) Compare both sid and sname values

```
public boolean equals(Object obj) {
  boolean flag=false;
  if (obj instanceof Student) {
     Student st = (Student) obj;
     flag = (this.sid == st.sid) &&
            this.sname.equals(st.sname);

  }
  return flag;
}
```

Assignment # 15

Q1) What is the difference between custom and built-in package?

Q2) What package is imported in java classes automatically?

Q3) What is the super class of Object class?

Q4) What is the default super class for all the Java classes?

Q5) What are the methods available in Object class?

Q6) What are the methods of Object class declared as protected?

Q7) What is the return type of getClass() method?

Q8) Can I override getClass() method?

Q9) What is hash code?

Q10) How can I get hash code of any object?

Q11) In which condition I should override hashCode() method?

Q12) What will happen when I print reference variable?

Q13) What is the return type of toString() method?

Q14) What is the default implementation of toString() method?

Q15) Can I override toString() method?

Q16) What is the real time use of toString() method?

Q17) What is the default implementation of equals() method?

Q18) What is the difference between == operator and equals() method of Object class?

Q19) What are JVM memory types?

Q20) What is the use of Method Area?

Q21) What is the use of Stack memory?

Q22) What is the use of Heap memory?

Q23) What is the reason for OutOfMemoryError?

Q24) What is the reason for StackOverflowError?

Q25) How JVM identifies the dead objects of your program?

Q26) What is the use of finalize() method?

Q27) Can I force GC?

Practice Test # 15

Q.No	Question	Options	Answer

1	public class Test {
	public static void main(String[] args) {
	Student st = new Student();
	Class cl =st.getClass();
	String name=cl.getName();
	System.out.println(name);
	}
	}
	
	class Student{}
		A) Student
		B) String
		C) java.lang.String
		D) Compilation Error
		E) Runtime Exception
		F) We can't decide the result	

2	public class Test {
	public static void main(String[] args) {
	Student st = new Student();
	Class cl =st.getClass();
	String name=cl.getName();
	System.out.println(name);
	}
	}
	
	class Student{}
		A) Student
		B) String
		C) java.lang.String
		D) Compilation Error
		E) Runtime Exception
		F) We can't decide the result	

3	public class Test {
	public static void main(String[] args) {
	Student st = new Student();
	System.out.println(st.hashCode());
	}
	}
	
	class Student{}
		A) 2145874
		B) 7825414
		C) Compilation Error
		D) Runtime Exception
		E) We can't decide the result	

4	public class Test {
	public static void main(String[] args) {
	Student st = new Student();
	System.out.println(st.hashCode());
	}
	}
	class Student{
	public int hashCode() {
	return 7825414;
	}
	}
		A) 2145874
		B) 7825414
		C) Compilation Error
		D) Runtime Exception
		E) We can't decide the result	

5	public class Test {
	public static void main(String[] args) {
	String str="Hello";
	System.out.println(str);
	}
	}
		A) Hello
		B) Compilation Error
		C) Runtime Exception	

6	public class Test {
	public static void main(String[] args) {
	Student st=new Student();
	System.out.println(st);
	}
	}
	
	class Student{ }
		A) Student@...
		B) JLCStudent
		C) Compilation Error
		D) Runtime Exception	

7	public class Test {
	public static void main(String[] args) {
	Student st=new Student();
	System.out.println(st);
	}
	}
	
	class Student{
	public String toString() {
	return "JLCStudent";
	}
	}
		A) Student@...
		B) JLCStudent
		C) Compilation Error
		D) Runtime Exception	

8	public class Test {
	public static void main(String[] args) {
	Student st=new Student();
	System.out.println(st);
	}
	}
	
	class Student{
	public int hashCode() {
	return 15;
	}
	}
		A) Student@15
		B) 15
		C) Student@f
		D) Student@....
		E) Compilation Error	

9	public class Test {
	public static void main(String[] args) {
	String s1=new String("JLC");
	String s2=new String("JLC");
	System.out.println(s1==s2);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

10	public class Test {
	public static void main(String[] args) {
	String s1=new String("JLC");
	String s2=new String("JLC");
	System.out.println(s1.equals(s2));
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

11	public class Test {
	public static void main(String[] args) {
	Student s1=new Student();
	Student s2=new Student();
	System.out.println(s1==s2);
	}
	}
	
	class Student{}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

12	public class Test {
	public static void main(String[] args) {
	Student s1=new Student();
	Student s2=new Student();
	System.out.println(s1.equals(s2));
	}
	}
	
	class Student{}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

13	public class Test {
	public static void main(String[] args) {
	Student s1 = new Student(99);
	Student s2 = new Student(99);
	System.out.println(s1.equals(s2));
	}
	}
	class Student {
	int sid;
	Student(int sid) {
	this.sid = sid;
	}
	public boolean equals(Student st) {
	return this.sid ==st.sid;
	} }
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

14	public class Test {
	public static void main(String[] args) {
	Student s1 = new Student("Sri");
	Student s2 = new Student("Sri");
	System.out.println(s1.equals(s2));
	}
	}
	class Student {
	String name;
	Student(String name) {
	this.name=name;
	}
	public boolean equals(Student st) {
	return this.name ==st.name;
	} }
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

15	public class Test {
	public static void main(String[] args) {
	Student s1 = new Student("Sri");
	Student s2 = new Student("Sri");
	System.out.println(s1.equals("Sri"));
	}
	}
	class Student {
	String name;
	Student(String name) {
	this.name=name;
	}
	public boolean equals(Student st) {
	return this.name ==st.name;
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

16	public class Test {
	public static void main(String[] args) {
	String nm1=new String("Sri");
	String nm2=new String("Sri");
	Student s1 = new Student(nm1);
	Student s2 = new Student(nm2);
	System.out.println(s1.equals(s2));
	}
	}
	
	class Student {
	
	String name;
	
	Student(String name) {
	this.name=name;
	}
	
	public boolean equals(Student st) {
	return this.name ==st.name;
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

17	public class Test {
	public static void main(String[] args) {
	String nm1=new String("Sri");
	String nm2=new String("Sri");
	Student s1 = new Student(nm1);
	Student s2 = new Student(nm2);
	System.out.println(s1.equals(s2));
	}
	}
	
	class Student {
	
	String name;
	
	Student(String name) {
	this.name=name;
	}
	
	public boolean equals(Student st) {
	return this.name.equals(st.name);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

18	public class Test {
	public static void main(String[] args) {
	String nm1=new String("Sri");
	String nm2=new String("Sri");
	Student s1 = new Student(nm1);
	Student s2 = new Student(nm2);
	System.out.println(s1.equals(s2));
	}
	}
	class Student {
	String name;
	Student(String name) {
	this.name=name;
	}
	public boolean equals(Object st) {
	return this.name.equals(st.name);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

19	public class Test {
	public static void main(String[] args) {
	new Hello();
	System.out.println("OK");
	} }
	class Hello {
	protected void finalize() throws Throwable {
	System.out.println("Hello -> finalize()");
	} }
		A) Hello -> finalize()
		OK
		B) OK
		Hello -> finalize()
		C) OK
		D) Compilation Error
		E) Runtime Error	

20	public class Test {
	public static void main(String[] args) {
	new Hello();
	System.out.println("OK");
	System.gc();
	}
	}
	class Hello {
	protected void finalize() throws Throwable {
	System.out.println("Hello -> finalize()");
	}
	}
		A) Hello -> finalize()
		OK
		B) OK
		Hello -> finalize()
		C) OK
		D) Compilation Error
		E) Runtime Error	

4.2.5 String Class

- String is a built-in class in java.lang package.
- String is final class, so you can't define the subclass for String class.
- String class implements the following interfaces:
  - java.io.Serializable
  - java.lang.Comparable
  - java.lang.CharSequence
- String class has following variable to hold data.

  private final char value[];
- Following is the String class definition by Java Vendor:

```
public final class String implements Serializable, Comparable, CharSequence{
    private final char value[];
  ...
}
```

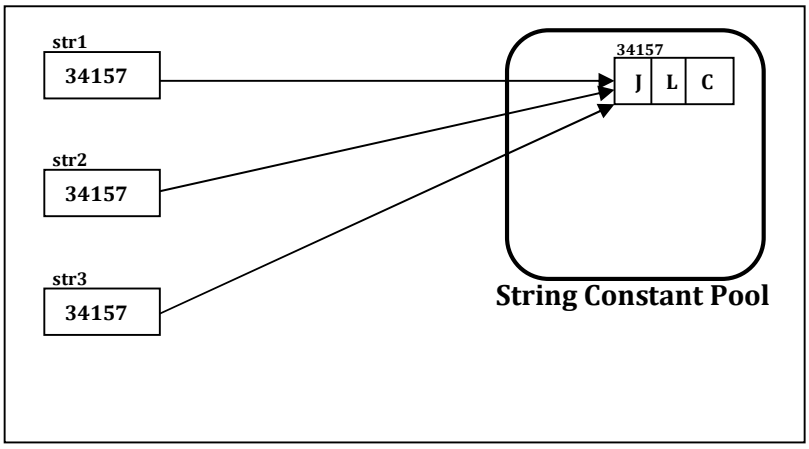
- String objects are immutable objects. It means once the object is created then the content or data of the object can't be modified.
- When you try to modify the contents of object then new object will be created as a result.
- You can create the object of String in two ways.
  - Without using new operator.
  - Using new operator.

4.2.5.1 Creating String object without new operator

1) JVM allocates the memory for the String reference variable.
2) JVM verifies the String literal in the String Constant Pool.
3) If String literal is not available in the String Constant Pool then it creates new String object inside the String Constant pool and newly created String object address will be assigned to String reference variable.
4) If String literal is available then existing String object address will be assigned to String reference variable.

Lab527.java

```
class Lab527{
public static void main(String as[]){
String str1="JLC";
String str2="JLC";
String str3="JLC";
System.out.println(str1==str2);
System.out.println(str1==str3);
System.out.println(str2==str3);
}
}
```

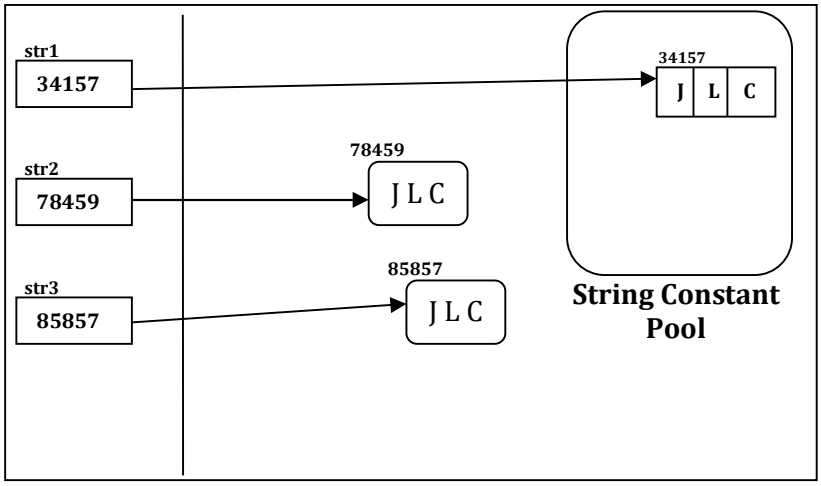


4.2.5.2 Creating String object with new operator

1) JVM allocates the memory for the String reference variable.
2) JVM verifies the String literal in the String Constant Pool.
3) If String literal is not available in the String Constant Pool then it creates new String object inside the String Constant pool.
4) If String literal is available in the String Constant Pool then ignores that.
5) It creates another new String object outside the constant pool and assigns address of the newly created String object outside the pool to String reference variable.

Lab528.java

```
class Lab528{
public static void main(String as[]){
String str1="JLC";
String str2=new String("JLC");
String str3=new String("JLC");
System.out.println(str1==str2);
System.out.println(str1==str3);
System.out.println(str2==str3);
}
}
```



4.2.5.3 String Class API

Important Constructors from String Class	
String()	Create a new String with the empty content
String(String str)	Create a new String with the specified content
String(char[] arr)	Create a new String with the specified array of chars
String(char[], int stIndex, int len)	Create a new String with the specified part of array of chars
String(byte[], int stIndex, int len)	Create a new String with the specified part of array of chars
String(byte[])	Create a new String with the specified array of bytes
String(StringBuffer)	Create a new String with the specified StringBuffer
String(StringBuilder)	Create a new String with the specified StringBuilder

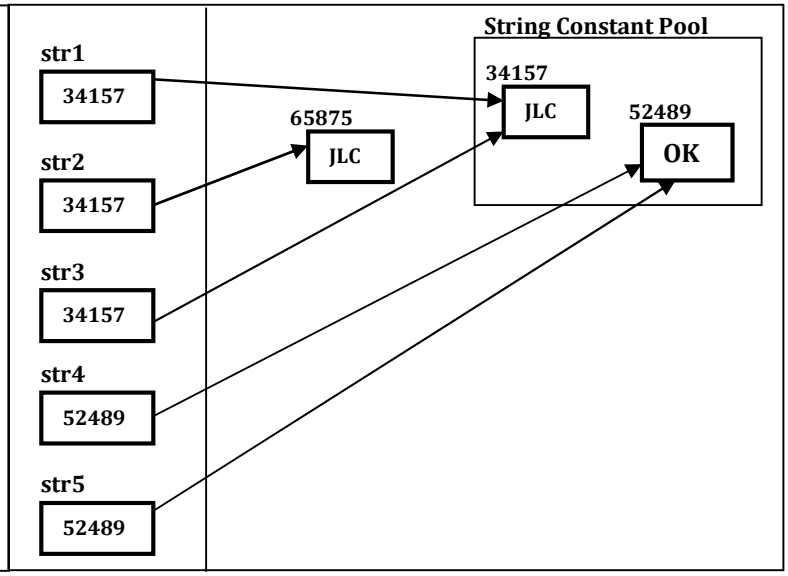
Important Methods from String Class	
native String intern()	Returns the reference of string object from Constant pool.
int length()	Returns the length of the String object.
boolean isEmpty()	Returns true if, and only if, length() is 0.
String concat(String)	Concatenates the specified string to the end of current string.
String toLowerCase()	Converts all of the characters in this String to lower case.
String toUpperCase()	Converts all of the characters in this String to upper case.
String trim()	Trim whitespace from the beginning and end of a string.
String toString()	Returns the current object content.
static String valueOf(X val) X can be Object,int,boolean,double etc	Returns the string representation of the Specified argument.
char charAt(int index)	Returns the char value at the specified index.
char[] toCharArray()	Converts this string to a new character array.
void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Copies characters from this string into the destination character array.
byte[] getBytes()	Converts this string to a new byte array using ASCII.
void getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)	Copies ASCII of the characters from this string into the destination byte array.
public boolean equals(Object)	Compares content of this string to the specified object Case of the character also will be verified.
boolean equalsIgnoreCase(String)	Compares content of this string to the specified object Case of the character will be ignored.
Boolean contentEquals(StringBuffer)	Compares content of this string to the specified StringBuffer object. Case of the character also will be verified.

int compareTo(String)	Compares content of this string to the specified object Case of the character also will be verified.
int compareToIgnoreCase(String)	Compares content of this string to the specified object Case of the character will be ignored.
boolean startsWith(String prefix)	Checks whether string starts with the specified prefix.
boolean startsWith(String p, int fromIdx)	Checks whether string starts with the specified prefix from specified starting index.
boolean endsWith(String suffix)	Checks whether string starts with the specified suffix.
int indexOf(int ch)	Returns the index of the first occurrence of the specified character.
int indexOf(int ch, int frmIndex)	Returns the index of the first occurrence of the specified character. It starts search from the specified index.
int indexOf(String st)	Returns the index of the first occurrence of the specified String.
int indexOf(String st, int frmIdx)	Returns the index of the first occurrence of the specified String. It starts search from the specified index.
int lastIndexOf(int ch)	Returns the index of the last occurrence of the specified character.
int lastIndexOf(int ch, int frmIdx)	Returns the index of the last occurrence of the specified character. It starts search backward from the specified index.
int lastIndexOf(String st)	Returns the index of the last occurrence of the specified String.
int lastIndexOf(String st, int frmIdx)	Returns the index of the last occurrence of the specified String. It starts search backward from the specified index.
String substring(int beginIdx)	Returns the part of the String. Substring begins with the character at the specified index and up to the end of this string.
String substring(int beginIdx, int endIdx)	Returns the part of the String. Substring begins with the character at the specified index and up to the (endIndex – 1).
String replace(char oChar, char nChar)	Replaces all occurrences of oChar with nChar.

String replaceFirst(String regEx, String newString)	Replaces first occurrences of matching String with newStrring.
String replaceAll(String regEx, String newString)	Replaces all occurrences of matching String with newStrring.
boolean matches(String regExpr)	Checks whether this string matches with the given regular expression.
String[] split(String regEx)	Splits this string when the given regular expression matches.
String[] split(String regEx, int limit)	Splits this string when the given regular expression matches. limit controls the number of times the pattern is applied and it affects the length of the resulting array.
static String format(String, Object...)	Returns a formatted string using the specified format string and arguments.
int hashCode()	Returns the hashCode of String object. Uses contents of String to generate the hashCode.

Lab529.java

```
public class Lab459{
public static void main(String[] args) {
String str1 = "JLC";
String srt2 = new String("JLC");
String str3 = st2.intern();
System.out.println(str1 == str2);
System.out.println(str1 == str3);
System.out.println(str2 == str3);
System.out.println();
String str4 = "OK".intern();
String str5 = "OK";
System.out.println(str4 == str5);
}
}
```



Lab530.java

```
class Lab530{
public static void main(String as[]){
String str1=new String("Hello");
String str2=new String(" Guys");
String str3=str1+str2;
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab531.java

```
cclass Lab531{
public static void main(String as[]){
String str1="Hello";
String str2=" Guys";
String str3=str1+str2;
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab532.java

```
class Lab532{
public static void main(String as[]){
String str1="Hello";
String str3=str1+" Guys";
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab533.java

```
class Lab533{
public static void main(String as[]){
String str2=" Guys";
String str3="Hello"+str2;
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab534.java

```
class Lab534{
public static void main(String as[]){
String str3="Hello"+" Guys";
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab535.java

```
class Lab535{
public static void main(String as[]){
String str3="Hello".concat(" Guys");
String str4="Hello Guys";
System.out.println(str3);
System.out.println(str4);
System.out.println(str3==str4);
}
}
```

Lab536.java

```
class Lab536{
public static void main(String as[]){
String str1=new String("JLC");
String str2=new String("JLC");
String str3=new String("MyJLC");
System.out.println(str1==str2);
System.out.println(str2==str3);
System.out.println(str1==str3);

System.out.println(str1.equals(str2));
System.out.println(str2.equals(str3));
System.out.println(str1.equals(str3));
}
}
```

Lab537.java

```
class Lab537{
public static void main(String as[]){

String str1=new String("JLC");
String str2=new String("jlc");

boolean b1=str1.equals(str2);
System.out.println(b1);
boolean b2=str1.equalsIgnoreCase(str2);
System.out.println(b2);

}
}
```

Lab538.java

```
class Lab538{
public static void main(String as[]){
String str1=new String("JLC");
int x =str1.compareTo("JLC");
System.out.println(x);

x =str1.compareTo("JLH");
System.out.println(x);

x =str1.compareTo("JLA");
System.out.println(x);
x =str1.compareTo("jlc");
System.out.println(x);
}
}
```

Lab539.java

```
class Lab539{
public static void main(String as[]){
String str1="A";
String str2="B";

int x =str1.compareTo(str2);
System.out.println(x);
String str3="B";
String str4="A";
x =str3.compareTo(str4);
System.out.println(x);
x ="A".compareTo("A");
System.out.println(x);
}
}
```

Lab540.java

```
class Lab540{
public static void main(String as[]){
String str1=new String("JLC");
String str2=new String("JLC");
String str3=new String("jlc");
int x =str1.compareTo(str2);
System.out.println(x);
x =str1.compareTo(str3);
System.out.println(x);
x =str1.compareToIgnoreCase(str3);
System.out.println(x);
}
}
```

Lab541.java

```
class Lab541{
public static void main(String as[]){
String str1="Srinivas";
String str2="";
System.out.println(str1.length());
System.out.println(str2.length());

System.out.println(str1.isEmpty());
System.out.println(str2.isEmpty());
}
}
```

Lab542.java

```
class Lab542{
public static void main(String as[]){
String str1="SriNiVas";
String str2="jlc";
String str3="JLC";
System.out.println(str1);
System.out.println(str1.toUpperCase());
System.out.println(str1.toLowerCase());
System.out.println(str1);
System.out.println(str2.toUpperCase());
System.out.println(str3.toLowerCase());
}
}
```

Lab543.java

```
class Lab543{
public static void main(String as[]){
//String str=99;
String str1=String.valueOf(99);
System.out.println(str1);
String str2=String.valueOf(99.99);
System.out.println(str2);
String str3=String.valueOf(true);
System.out.println(str3);
String str4=String.valueOf(999L);
System.out.println(str4);
String str5=String.valueOf('A');
System.out.println(str5);
}
}
```

Lab544.java

```
class Lab544{
public static void main(String as[]){
char chArr[]={'S','R','I','N','I','V','A','S',' ','D'};
String str1=String.valueOf(chArr);
System.out.println(str1);
String str2=String.valueOf(chArr,3,5);
System.out.println(str2);
String str3=String.valueOf(chArr,5,3);
System.out.println(str3);

}
}
```

Lab545.java

```
class Customer{
public String toString(){
return "I am Customer";
}
}
class Account{}

class Lab545{
public static void main(String as[]){
Customer cust=new Customer();
Account acc=new Account();
String str1=String.valueOf(cust);
System.out.println(str1);
String str2=String.valueOf(acc);
System.out.println(str2);
}
}
```

Lab546.java

```
class Lab546{
public static void main(String as[]){
String str2=String.valueOf(null);
System.out.println(str2);
}
}
```

Lab547.java

```
class Hello{}

class Lab547{
public static void main(String as[]){
Hello hello=null;
String str1=String.valueOf(hello);
System.out.println(str1);
}
}
```

Lab548.java

```
class Lab548{
public static void main(String as[]){
String str="Hello Guys";

boolean b1 = str.startsWith("Hello");
System.out.println(b1);

b1 = str.startsWith("Hai");
System.out.println(b1);
b1 = str.startsWith("Guys");
System.out.println(b1);
b1 = str.startsWith("Guys",6);
System.out.println(b1);
b1 = str.startsWith("Hello",5);
System.out.println(b1);
}
}
```

Lab549.java

```
class Lab549{
public static void main(String as[]){
String str="Hello Guys";

boolean b1 = str.endsWith("Hello");
System.out.println(b1);

b1 = str.endsWith("Guys");
System.out.println(b1);
b1 = str.endsWith("ys");
System.out.println(b1);
b1 = str.endsWith("sd");
System.out.println(b1);

}
}
```

Lab450.java

```
class Lab550{
public static void main(String as[]){

String str="Hello Guys";
boolean b1=str.startsWith("");
System.out.println(b1);
boolean b2= str.endsWith("");
System.out.println(b2);
}
}
```

Lab451.java

```
class Lab551{
public static void main(String as[]){

String str="Hello Guys";
System.out.println(str.charAt(0));
System.out.println(str.charAt(4));
System.out.println(str.charAt(6));
//System.out.println(str.charAt(10));
}
}
```

Lab552.java

```
class Lab552{
public static void main(String as[]){

String str="Hello Guys";

System.out.println(str.indexOf('H'));
System.out.println(str.indexOf('O'));
System.out.println(str.indexOf('o'));
System.out.println(str.indexOf('G'));
System.out.println(str.indexOf('Z'));
}
}
```

Lab553.java

```
class Lab553{
public static void main(String as[]){

String str="Hello Guys Hey Guys";

System.out.println(str.indexOf("Guys"));
System.out.println(str.indexOf("He"));
System.out.println(str.indexOf("Ha"));

System.out.println(str.indexOf("Guys",7));
System.out.println(str.indexOf("He",5));
System.out.println(str.indexOf("Ha",9));
}
}
```

Lab554.java

```
class Lab554{
public static void main(String as[]){

String str="srinivas";

System.out.println(str.indexOf('i'));
System.out.println(str.lastIndexOf('i'));

System.out.println(str.indexOf('s'));
System.out.println(str.lastIndexOf('s'));

System.out.println(str.indexOf('n'));
System.out.println(str.lastIndexOf('n'));
}
}
```

Lab555.java

```
class Lab555{
public static void main(String as[]){

String str="Hello Guys Hey Guys";

System.out.println(str.lastIndexOf("Guys"));
System.out.println(str.lastIndexOf("He"));
System.out.println(str.lastIndexOf("Ha"));

System.out.println(str.lastIndexOf("Guys",10));
System.out.println(str.lastIndexOf("He",10));
System.out.println(str.lastIndexOf("Ha",10));
}
}
```

Lab556.java

```
class Lab556{
public static void main(String as[]){

String str="Hello Guys How are you";

String str1 = str.substring(11);
System.out.println(str1);

String str2 = str.substring(6,10);
System.out.println(str2);

String str3 = str.substring(0,5);
System.out.println(str3);

String str4 = str.substring(19);
System.out.println(str4);

String str5 = str.substring(19,22);
System.out.println(str5);

}
}
```

Lab557.java

```
class Lab557{
public static void main(String as[]){

String str="   Hello   Guys   ";
System.out.println(str);
System.out.println(str.length());

String str1=str.trim();

System.out.println(str1);
System.out.println(str1.length());
}
}
```

Lab558.java

```
class Lab558{
public static void main(String as[]){

String str="Srinivas";

String str1=str.replace('i','I');
System.out.println(str1);

String str2=str.replace("n","N");
System.out.println(str2);

String str3=str.replace("ni"," Hello ");
System.out.println(str3);

String str4=str3.replace(" Hello ","ni");
System.out.println(str4);
}
}
```

Lab559.java

```
class Lab559{
public static void main(String as[]){

String str="srinivas";

String str1=str.replaceFirst("i","I");
System.out.println(str1);

String str2=str.replaceAll("s","S");
System.out.println(str2);

String str3=str.replace("i","I");
System.out.println(str3);

}
}
```

Lab560.java

```
class Lab560{
public static void main(String as[]){

String str="Srinivas";

char chArr[] = str.toCharArray();
for(char ch:chArr)
System.out.println(ch);

byte byArr[] = str.getBytes();
for(byte by :byArr)
System.out.println(by);
}
}
```

Lab561.java

```
class Lab561{
public static void main(String as[]){

String str="Srinivas";
char destArr[] = new char[10];
destArr[0]='J';
destArr[1]='L';
destArr[2]='C';

str.getChars(5,8,destArr,4);
for(char ch:destArr)
System.out.println(ch);
}
}
```

Lab562.java

```
class Lab562{
public static void main(String as[]){

String str="Srinivas";
byte destArr[] = new byte[10];
destArr[0]=65;
destArr[1]=66;

str.getBytes(0,3,destArr,3);
for(byte by:destArr)
System.out.println(by);

}
}
```

Lab563.java

```
class Lab563{
public static void main(String as[]){

String str="Hello Guys How are you Guys";

String strArr1[] = str.split(" ");
System.out.println(strArr1.length);
for(String x:strArr1)
System.out.println(x);

String strArr2[] = str.split(" ",3);
System.out.println(strArr2.length);
for(String x:strArr2)
System.out.println(x);
}
}
```

Lab564.java

```
class Lab564{
public static void main(String as[]){

String str="Hello Guys How are you Guys Ok Ok";

String strArr1[] = str.split("Guys");
System.out.println(strArr1.length);
for(String x:strArr1)
System.out.println(x);

}
}
```

Lab565.java

```
class Lab565{
public static void main(String as[]){

String str="Hello Guys Ok Guys Ok";

boolean b1= str.contains("OK");
System.out.println(b1);
boolean b2= str.contains("Ok");
System.out.println(b2);
boolean b3= str.contains("Hello");
System.out.println(b3);
}
}
```

Lab566.java

```
class Lab566{
public static void main(String as[]){

String str1= new String();
System.out.println(str1.length());

String str2= new String("JLC");
System.out.println(str2.length());

}
}
```

Lab567.java

```
class Lab567{
public static void main(String as[]){

char chArr[] = {'S','R','I','N','I','V','A','S'};

String str1= new String(chArr);
System.out.println(str1);
System.out.println(str1.length());

String str2= new String(chArr,3,5);
System.out.println(str2);
System.out.println(str2.length());
}
}
```

Lab568.java

```
class Lab568{
public static void main(String as[]){

byte byArr[] = {65,66,67,68,69,70,71,72};

String str1= new String(byArr);
System.out.println(str1);
System.out.println(str1.length()); //

String str2= new String(byArr,1,6);
System.out.println(str2);
System.out.println(str2.length());
}
}
```

Lab569.java

```
class Lab569{
public static void main(String as[]){

String str1="A";

int hash1=str1.hashCode();
System.out.println(hash1);

String str2="AB";
int hash2=str2.hashCode();
System.out.println(hash2);
}
}
```

Lab570.java

```
class Lab570{
public static void main(String as[]){
int a=10;
int b=20;
int c=a+b;
String str1=String.format("Sum of %d and %d is
%d",a,b,c);
String str2="Sum of "+a+" and "+b+" is "+c;
System.out.println(str1);
System.out.println(str2);
}
}
```

Lab571.java

```
class Lab571{
public static void main(String as[]){
String exp1="[A-Z]";

String str="JLC";
boolean b=str.matches(exp1);
System.out.println(b);

System.out.println("J".matches(exp1));
System.out.println("a".matches(exp1));
System.out.println("A9".matches(exp1));

}
}
```

Lab572.java

```
class Lab572{
public static void main(String as[]){
String exp2="[A-Z]*";

System.out.println("JLC".matches(exp2));
System.out.println("J".matches(exp2));
System.out.println("".matches(exp2));
System.out.println("jlc".matches(exp2));
System.out.println("jlc9".matches(exp2));

}
}
```

Lab573.java

```
class Lab573{
public static void main(String as[]){
String exp3="[A-Za-z]*";

System.out.println("JLC".matches(exp3));
System.out.println("J".matches(exp3));
System.out.println("".matches(exp3));
System.out.println("jlc".matches(exp3));
System.out.println("MyJlc".matches(exp3));
System.out.println("Jlc9".matches(exp3));

}
}
```

Lab574.java

```
class Lab574{
public static void main(String as[]){
String exp4="[A-Za-z0-9]*";

System.out.println("JLC99".matches(exp4));
System.out.println("99".matches(exp4));
System.out.println("".matches(exp4));
System.out.println("jlc".matches(exp4));
System.out.println("MyJlc".matches(exp4));
System.out.println("Jlc9".matches(exp4));
System.out.println("Jlc$9".matches(exp4));
}
}
```

Lab575.java

```
class Lab575{
public static void main(String as[]){
String exp5="[A-Za-z0-9#@$]*";

System.out.println("JLC*99".matches(exp5));
System.out.println("99".matches(exp5));
System.out.println("".matches(exp5));
System.out.println("jlc".matches(exp5));
System.out.println("MyJlc".matches(exp5));
System.out.println("Jlc9".matches(exp5));
System.out.println("Jlc$9".matches(exp5));
System.out.println("Jlc#@99".matches(exp5));
}
}
```

Lab576.java

```
//Name Pattern
class Lab576{
public static void main(String as[]){
String exp6="[A-Z][A-Za-z ]*";

System.out.println("".matches(exp6));
System.out.println("sri nivas".matches(exp6));
System.out.println("Sri nivas".matches(exp6));
System.out.println("Sri Ni Vas".matches(exp6));
System.out.println("Sri
nivas99".matches(exp6));

}
}
```

Lab577.java

```
// Email Pattern
class Lab577{
public static void main(String as[]){
String exp7="[A-Za-z0-9]*[@][A-Za-z]*[.][A-Za-
z]*";

System.out.println("".matches(exp7));
System.out.println("@.".matches(exp7));
System.out.println("Sri@myjlc.com".matches(ex
p7));
System.out.println("s@s.com".matches(exp7));

}
}
```

Lab578.java

```
// Email Pattern
class Lab578{
public static void main(String as[]){
String exp8="[a-z][a-z][a-z]*[@][a-z][a-z][a-
z]*[.][a-z][a-z][a-z]*";

System.out.println("".matches(exp8));
System.out.println("@.".matches(exp8));
System.out.println("s@s.com".matches(exp8));
System.out.println("Sri@myjlc.d".matches(exp8
));
System.out.println("sd@sd.in".matches(exp8));
System.out.println("sri@myjlc.com".matches(ex
p8));
}
}
```

Lab579.java

```
// Phone Pattern
class Lab579{
public static void main(String as[]){
String exp9="[6-9][0-9]*";

System.out.println("jlc99".matches(exp9));
System.out.println("".matches(exp9));
System.out.println("3".matches(exp9));
System.out.println("123456".matches(exp9));
System.out.println("6".matches(exp9));
System.out.println("923456".matches(exp9));
}
}
```

Lab580.java

```
// Phone Pattern
class Lab580{
public static void main(String as[]){
String exp10="[+91-]*[6-9][0-9]*";

System.out.println("123456".matches(exp10));
System.out.println("+91-
123456".matches(exp10));
System.out.println("+91-
923456".matches(exp10));
System.out.println("923456".matches(exp10));
}
}
```

4.2.6 StringBuffer Class

- StringBuffer is a final class in java.lang package.
- It is used to store the sequence of characters.
- This is used when there is a necessity to make a lot of modifications to characters of String.
- StringBuffer is a mutable sequence of characters, It means the contents of the StringBuffer can be modified after creation.
- Every StringBuffer object has a capacity associated with it.
- The capacity of the StringBuffer is the number of characters it can hold.
- The capacity increases automatically as more contents added to it.
- The methods available in the StringBuffer class are synchronized.
- StringBuffer class is thread-safe i.e. multiple threads cannot access it simultaneously.
- So to solve this problem Java vendor has added a new class StringBuilder in Java 5.

4.2.7 StringBuilder Class

- StringBuilder is a final class in java.lang package.
- StringBuilder is a newly added class from Java 5.
- StringBuilder class functionality is same as StringBuffer only except the methods available in the StringBuilder class are non synchronized.
- StringBuilder class is not thread-safe i.e. multiple threads can access it simultaneously.

4.2.7.1 StringBuilder Class API

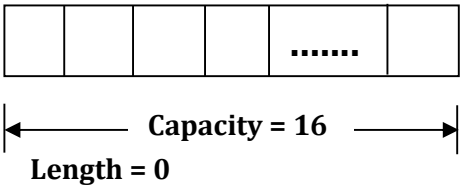
Importants methods of StringBuilder	
Methods	**Description**
int capacity()	Returns the capacity of StringBuilder object
int length()	Returns the length of StringBuilder object
void ensureCapacity(int cap)	Ensures the Capacity of the StringBuilder Object The new capacity of the StringBuilder is the maximum of, • The initialCapacity argument passed and • (Old capacity * 2) + 2

void setLength(int len)	Changes the length of StringBuilder • If new length is more than current length then default value of char (ASCII 0) will be added.
void trimToSize()	Reduces the capacity to length
StringBuilder append(X value)	Adds the data in the StringBuilder object • Method is overloaded • X can be any primitive type/String/Object
StringBuilder insert(int index, X value)	Inserts the data at specified index • Method is overloaded • X can be any primitive type/String/Object
StringBuilder deleteCharAt(int index)	Deletes the char from the specified index
StringBuilder delete (int index1,int index2)	Deletes the character from the specified range
void setCharAt(int index,char newChar)	Replaces char at the specific index
StringBuilder replace(int idx1,int idx2, String newValue)	Replaces the chars of Specific range with new String
StringBuilder reverse()	Reverse the content of StringBuilder
etc	

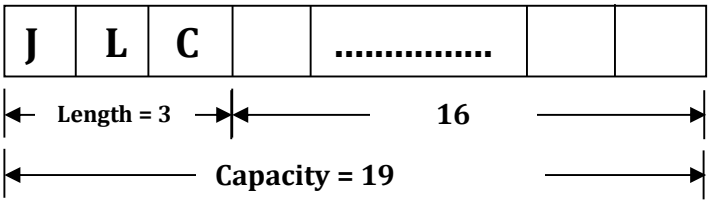
Constructors from StringBuilder Class	
public StringBuilder ()	Creates the StringBuilder object with INITIAL CAPACITY 16.
public StringBuilder (String content)	Creates the StringBuilder object with INITIAL CAPACITY as (length of specified content + 16).
public StringBuilder (int capacity)	Creates the StringBuilder object with INITIAL CAPACITY as specified

- equals() is not overriden in StringBuilder and StringBuffer class.
- When you want to compare contents of StringBuilder or StringBuffer object then you need to convert the object of StringBuilder or StringBuffer into String.
- hashCode() is not overriden in the StringBuilder and StringBuffer class. It uses the default implementation of Object class.
- StringBuilder and StringBuffer are not implementing java.lang.Comparable interface, So You can't use compareTo() with StringBuilder and StringBuffer.

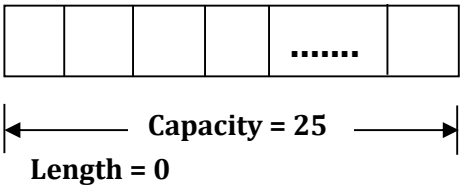
StringBuilder sb=new StringBuilder();



StringBuilder sb=new StringBuilder("JLC");



StringBuilder sb=new StringBuilder(25);



Lab581.java

```
class Lab581{
public static void main(String as[]){
String str1="JLC";
String str2=new String("JLC");
System.out.println(str1==str2);

//StringBuilder sb1="JLC";
StringBuilder sb2=new StringBuilder("JLC9");
StringBuffer sb3=new StringBuffer("JLC");
//System.out.println(sb2==sb3);

System.out.println(sb2.length());
System.out.println(sb2.capacity());

System.out.println(sb3.length());
System.out.println(sb3.capacity());
}
}
```

Lab582.java

```
class Lab582{
public static void main(String as[]){

StringBuilder sb1=new StringBuilder();
System.out.println(sb1.length());
System.out.println(sb1.capacity());

StringBuilder sb2=new StringBuilder("Hello");
System.out.println(sb2);
System.out.println(sb2.length());
System.out.println(sb2.capacity());

sb2.append(" Guys!!!");
System.out.println(sb2);
System.out.println(sb2.length());
System.out.println(sb2.capacity());

sb2.append("1234567899");
System.out.println(sb2);
System.out.println(sb2.length());
System.out.println(sb2.capacity());

sb2.trimToSize();
System.out.println(sb2);
System.out.println(sb2.length());
System.out.println(sb2.capacity());
}
}
```

Lab583.java

```
class Lab583{
public static void main(String as[]){

StringBuilder sb=new StringBuilder(25);
System.out.println(sb.length());
System.out.println(sb.capacity());

sb.append("Hello Guys!!!");
sb.append(true);
System.out.println(sb);
System.out.println(sb.length());
System.out.println(sb.capacity());

sb.insert(6,"JLC ");
System.out.println(sb);
System.out.println(sb.length());
sb.insert(10,true);
System.out.println(sb);
System.out.println(sb.length());
}
}
```

Lab584.java

```
class Lab584{
public static void main(String as[]){

StringBuilder sb=new StringBuilder("Hello JLC Guys!!! ");
System.out.println(sb);
System.out.println(sb.length());

sb.deleteCharAt(15);
sb.deleteCharAt(16);
System.out.println(sb);
System.out.println(sb.length());

sb.delete(9,15);
System.out.println(sb);
System.out.println(sb.length());

}
}
```

Lab585.java

```
class Lab585{
public static void main(String as[]){

StringBuilder sb=new StringBuilder("Hello Hai Guys!!! ");
System.out.println(sb);
System.out.println(sb.length());

sb.replace(6,9,"JLC");
System.out.println(sb);
System.out.println(sb.length());

sb.replace(5,18,"J");
System.out.println(sb);
System.out.println(sb.length());

sb.reverse();
System.out.println(sb);
System.out.println(sb.length());
}
}
```

Lab586.java

```
class Lab586{
public static void main(String as[]){

StringBuilder sb1=new StringBuilder("JLC");
StringBuilder sb2=new StringBuilder("JLC");

System.out.println(sb1==sb2);
System.out.println(sb1.equals(sb2));

//Option 1:
String str1=sb1.toString();
String str2=sb2.toString();
System.out.println(str1.equals(str2));

//Option 2:
String str=sb1.toString();
System.out.println(str.contentEquals(sb2));
}
}
```

Lab587.java

```
class Lab587{
public static void main(String as[]){

StringBuilder sb=new StringBuilder("JLC");

System.out.println(sb.hashCode());

}
}
```

4.2.7.2 Difference between String and StringBuffer

String Class	StringBuffer class
String objects are immutable.	StringBuffer objects are mutable.
We can create the object of String class using new operator or without new operator.	We need to use new operator to create the object of StringBuffer class.
String operations such as append would be less efficient if performed using String.	String operations such as append would be more efficient if performed using StringBuffer.
Methods are non synchronized. (It's object is not thread safe)	Methods are synchronized. (It's object is thread safe)
Implementing java.lang.Comparable interface.	Not implementing java.lang.Comparable interface
equals () is overriden to compare the contents.	equals () is not overriden to compare the contents.
Concatination Operator (+) can be used.	Concatination Operator (+) can't be used.
hashCode() is overridden using content of String.	hashCode() is not overridden.

SUMMARY

String Class

1) When you create two String objects without using new operator and with same content then both reference variables point to the same object.
2) When you create String object using new operator then every time one new object will be created outside the pool and that address is assigned to reference variable.
3) Consider the following case:

   String str1="JLC";
   String str2="JLC";
   Here both str1 and str2 point to same object.
4) Consider the following case:

   String str1="JLC";
   String str2=args[0]; (Provide JLC as Command Line Argument)
   Here both str1 and str2 point to different objects.

5) intern() method returns the reference of an object which is available in String Constant Pool. If String object is not available then it creates new object inside the pool and returns that newly created object reference.
6) If you are using String Concatenation operator then
    a. When the operands are literals or final variables then returns the reference of an object which is available in String Constant Pool.
    b. When any of the operand is non-final variable then it creates new object outside the pool and returns that newly created object reference.
7) Priority of Concatenation operator is higher than relational operator.

equals() method of String Class

8) Inherited from Object class and overridden in String class.
9) Returns true if and only if the argument
    Is not null
    Is a String object
    Represents the same sequence of characters as this object

compareTo() method of String Class

10) This method compares the String content by using ASCII value.
11) If ASCII difference is found then difference is returned otherwise length difference will be returned.
    If it is returning 0 (Zero) then both the objects are same.
    If it is returning +ve then first object is higher or greater.
    If it is returning -ve then second object is higher or greater.

valueOf(X val) method of String Class

12) When you pass primitive as parameter to valueOf() method then String representation of that primitive will be returned.
13) Following is the implementation of valueOf() method when you pass object as parameter

```
public static String valueOf(Object obj) {
    return (obj == null) ? "null" : obj.toString();
  }
```

split() method of String Class

public String[] split(String expr)
public String[] split(String expr, int limit)

14) When you are not specifying the limit then String will be splited using all the occurance of the expression.
15) When you are specifying the limit as 0 (Zero) or more than occurance of expression then String will be splited using all the occurance of the expression.
16) When you are specifying the limit as more than 0 (Zero) but less than occurance of expression then it will split that upto that limit only.

hashCode() of String Class

17) The following implementation provided for hashCode in String class:

s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]

String st="A";	String st="AB";
Len 1	Len 2
$65 * 31^{(1-1)}$	$65 * 31^{(2-1)} + 66 * 31^{(2-2)}$
$65 * 31^0$	$65 * 31^1 + 66 * 31^0$
65 * 1	65 * 31 + 66 * 1
65	2015 + 66
	2081

Assignment # 16

Q28) Can I write the subclass for String class?

Q29) What are super interfaces of String class?

Q30) What is the variable of String class to hold String contents?

Q31) What is immutable objects?

Q32) What happens when you modify the String objects?

Q33) What is the difference between followings:

Q34) Creating String object Without using new operator.

Q35) Creating String object using new operator.

Q36) What is the use of intern() method?

Q37) What are the methods of Object class overridden in String class?

Q38) What is String constant pool and what is its usage?

Q39) What is the hashCode() implementation of String class?

Q40) How can I prove that String class is immutable? Explain with an example.

Q41) What is the difference between equals() and equalsIngnoreCase() ?

Q42) What is the difference between campareTo() and compareToIngnoreCase() ?

Q43) What is the difference between equals() and campareTo() ?

Q44) What is the difference between StringBuffer and StringBuilder?

Q45) What is the difference between length() and capacity()of StringBuilder class?

Q46) How to Compare String and StringBuilder?

Q47) What is the method to reverse the String?

Q48) What is the method to reverse the String StringBuilder?

Q49) Can I concatenate String and StringBuilder using + operator?

Practice Test # 16

Q.No	Question	Options	Answer

1	public class Test {
	public static void main(String[] args) {
	String s1 = "JLC";
	String s2 = new String("JLC");
	System.out.println(s1 == s2);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

2	public class Test {
	public static void main(String[] args) {
	String s1 = "JLC";
	String s2 = "INDIA";
	String s3="JLCINDIA";
	System.out.println(s1+s2 == s3);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

3	public class Test {
	public static void main(String[] args) {
	final String s1 = "JLC";
	final String s2 = "INDIA";
	String s3="JLCINDIA";
	System.out.println(s1+s2 == s3);
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

4	public class Test {
	public static void main(String[] args) {
	String str = null;
	System.out.println(str.isEmpty());
	}
	}
		A) true
		B) false
		C) Compilation Error
		D) Runtime Error	

5	public class Test {
	public static void main(String[] args) {
	System.out.println(null.length());
	}
	}
		A) null
		B) 0
		C) Compilation Error
		D) Runtime Error	

6	public class Test {
	public static void main(String[] args) {
	String str = "null";
	System.out.println(str.length());
	}
	}
		A) 0
		B) 4
		C) Compilation Error
		D) Runtime Error	

7	public class Test {
	public static void main(String[] args) {
	StringBuilder sb1 = new StringBuilder(-3);
	System.out.println(sb1.capacity());
	}
	}
		A) -3
		B) 0
		C) Compilation Error
		D) Runtime Error	

8	public class Test {
	public static void main(String[] args) {
	StringBuilder sb1 = new StringBuilder(879898767);
	System.out.println(sb1.capacity());
	}
	}
		A) -3
		B) 0
		C) Compilation Error
		D) Runtime Error	

9	public class Test {
	public static void main(String[] args) {
	StringBuilder sb = new StringBuilder(null);
	System.out.println(sb);
	}
	}
		A) -3
		B) 0
		C) Compilation Error
		D) Runtime Error	

10	public class Test {
	public static void main(String[] args) {
	StringBuilder sb = new StringBuilder();
	sb.append(null);
	System.out.println(sb);
	}
	}
		A) -3
		B) 0
		C) Compilation Error
		D) Runtime Error	

11	public class Test {
	public static void main(String[] args) {
	String str = "JLC";
	StringBuilder sb = new StringBuilder("JLC");
	System.out.println(str.equals(sb));
	}
	}
		A) -3
		B) 0
		C) Compilation Error
		D) Runtime Error	

4.2.8 Wrapper Classes

- When you are developing application then you can use both primitive type and Object type value.
- Where primitive type is required object cannot be used and where object is required primitive cannot be used.

```
ArrayList al = new ArrayList();
al.add(123);            //Invalid upto Jdk1.4
Integer iref = new Integer(123);
                  // Primitive to Object
al.add(iref);           //Valid
…
Iterator it = al.Iterator();
while(it.hasNext()){
Object obj = it.next();
Integer in = (Integer)obj;
int x=in;               // Invalid
int x=in.intValue();    //Object to Primitive
}
```

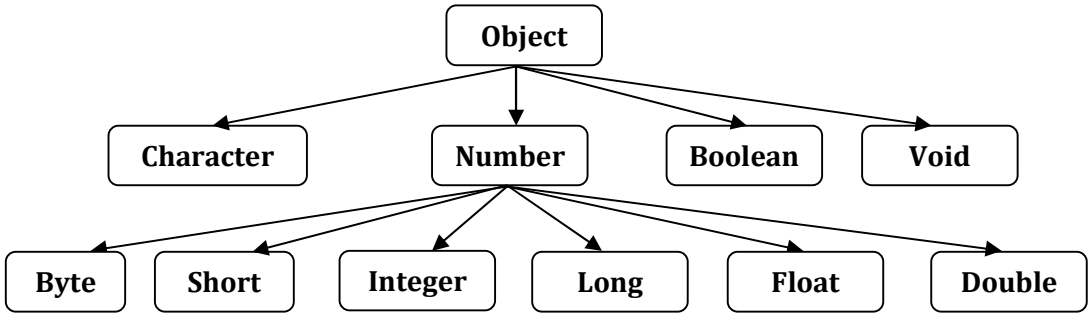
```
String str="123";
int x=str;// Not ok

int x=Integer.parseInt(str);
    //Converting one type to another type
```

- You can use Wrapper classes and their methods to convert:
  - Primitive to Object type.
  - Object to Primitive type.
  - One type to another type (Primitive, Wrapper, String).
  - One base to another base(Decimal, Binary, Octal, Hexadecimal)

- Following are Wrapper classes available in Java:

Primitive Data Type	Wrapper Class
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double
void	Void



4.2.8.1 Important Conversions using Wrapper Classes

1. Primitive to String

A) using ValueOf(primitive) methods of String
B) using toString(primitive) method of Wrapper class.

2. String to Primitive

A) using parseX();

3. Primitive to Wrapper Object

A) Using Constructor of Wrapper class
B) Using valueOf() method of Wrapper class

4. Wrapper Object to Primitive

A) Using xxValue() method of Wrapper classes

5. String to Wrapper Object

A) Using constrcutor of Wrapper classes
B) Using valueOf() method of Wrapper classes

6. Wrapper Object to String

A) Using toString() method

Lab588.java

```
class Lab588{
public static void main(String as[]){

int a=10;
String str1= String.valueOf(a);
System.out.println(str1);

double d=99.99;
String str2= String.valueOf(d);
System.out.println(str2);

boolean b= true;
String str3= String.valueOf(b);
System.out.println(str3);

}
}
```

Lab589.java

```
class Lab589{
public static void main(String as[]){

int a=10;
String str1= Integer.toString(a);
System.out.println(str1);

double d=99.99;
String str2= Double.toString(d);
System.out.println(str2);

boolean b= true;
String str3= Boolean.toString(b);
System.out.println(str3);

}
}
```

Lab590.java

```
class Lab590{
public static void main(String as[]){

String str1="123";

int a= Integer.parseInt(str1);
System.out.println(a);

String str2="99.99";
double d= Double.parseDouble(str2);
System.out.println(d);

String str3="true";
boolean b = Boolean.parseBoolean(str3);
System.out.println(b);

}
}
```

Lab591.java

```
class Lab591{
public static void main(String as[]){

int a=123;
Integer i1= new Integer(a);
System.out.println(i1);

double d=99.99;
Double d1=new Double(d);
System.out.println(d1);

boolean b=true;
Boolean b1 = new  Boolean(b);
System.out.println(b1);

}
}
```

Lab592.java

```
class Lab592{
public static void main(String as[]){

int a=123;
Integer i1= Integer.valueOf(a);
System.out.println(i1);

double d=99.99;
Double d1=Double.valueOf(d);
System.out.println(d1);

boolean b=true;
Boolean b1 = Boolean.valueOf(b);
System.out.println(b1);

}
}
```

Lab593.java

```
class Lab593{
public static void main(String as[]){

Integer i1= new Integer(123);
byte x=i1.byteValue();
short s=i1.shortValue();
int a=i1.intValue();
long y=i1.longValue();
float f=i1.floatValue();
double z=i1.doubleValue();

System.out.println(x);
System.out.println(s);
System.out.println(a);
System.out.println(y);
System.out.println(f);
System.out.println(z);
}
}
```

Lab594.java

```
class Lab594{
public static void main(String as[]){

Double d=new Double(99.99);

byte x=d.byteValue();
short s=d.shortValue();
int a=d.intValue();
long y=d.longValue();
float f=d.floatValue();
double z=d.doubleValue();

System.out.println(x);
System.out.println(s);
System.out.println(a);
System.out.println(y);
System.out.println(f);
System.out.println(z);
}
}
```

Lab595.java

```
class Lab595{
public static void main(String as[]){

Boolean b1 = new Boolean(true);
boolean b=b1.booleanValue();
System.out.println(b);

Character ch1=new Character('A');
char ch=ch1.charValue();
System.out.println(ch);
}
}
```

Lab596.java

```
class Lab596{
public static void main(String as[]){

String str1="123";
Integer i1=new Integer(str1);
System.out.println(i1);

String str2="99.99";
Double d1=new Double(str2);
System.out.println(d1);

String str3="true";
Boolean b1=new Boolean(str3);
System.out.println(b1);

}
}
```

Lab597.java

```
class Lab597{
public static void main(String as[]){

String str1="A";
Character ch=new Character(str1);
System.out.println(ch);

}
}
```

Lab598.java

```
class Lab598{
public static void main(String as[]){

String str1="123";
Integer i1= Integer.valueOf(str1);
System.out.println(i1);

String str2="99.99";
Double d1=Double.valueOf(str2);
System.out.println(d1);

String str3="true";
Boolean b1=Boolean.valueOf(str3);
System.out.println(b1);
}
}
```

Lab599.java

```
class Lab599{
public static void main(String as[]){

Integer i1= new Integer(123);
String str1=i1.toString();
System.out.println(str1);

Double d1=new Double(99.99);
String str2=d1.toString();
System.out.println(str2);

Boolean b1=new Boolean(true);
String str3=b1.toString();
System.out.println(str3);
}
}
```

Lab600.java

```
class Lab600{
public static void main(String as[]){

String str="123Sri";
int a=Integer.parseInt(str);
System.out.println(a);
}
}
```

Lab601.java

```
class Lab601{
public static void main(String as[]){

String str="123Sri";
Integer i1=new Integer(str);
System.out.println(i1);
}
}
```

Lab602.java

```
class Lab602{
public static void main(String as[]){

String str1="True";
Boolean b1= new Boolean(str1);
System.out.println(b1);

String str2="Hello";
Boolean b2= new Boolean(str2);
System.out.println(b2);

}
}
```

Lab603.java

```
class Lab603{
public static void main(String as[]){

String str1="99.99J";
Double d1= new Double(str1);
System.out.println(d1);

}
}
```

Lab604.java

```
class Lab604{
public static void main(String as[]){

System.out.println(Byte.MIN_VALUE);
System.out.println(Byte.MAX_VALUE);

System.out.println(Short.MIN_VALUE);
System.out.println(Short.MAX_VALUE);

}
}
```

Lab604.java

```
class Lab604{
public static void main(String as[]){

System.out.println(Byte.MIN_VALUE);
System.out.println(Byte.MAX_VALUE);

System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE);

}
}
```

4.2.9. AutoBoxing

Before Java 5

```
int x=10;
Integer ref=x;  //Not Ok
Integer ref= new Integer(x);  // Boxing

Integer ref= new Integer(x);
int x=ref;  //Not Ok
int x=ref.intValue();  //Unboxing
```

Boxing:

- Boxing is the process of converting primitive value to wrapper object.

UnBoxing:

- UnBoxing is the process of converting wrapper object to primitive value.

- From JDK1.5 onwards these two tasks will be done by the Compiler implicitly called as AutoBoxing and AutoUnboxing process.

From Java 5

```
Integer ref1=123;                          // Valid -Autoboxing
Integer ref2=new Integer(123);             // Valid
                OR
Integer ref=new Integer(234);
int x=ref;                        // Valid   -Autounboxing
int y=ref.intValue();             // Valid
```

- You can create the wrapper class object without using new operator. Caching concept is used if you are creating object without new operator.

Lab605.java

```
class Lab605{
public static void main(String as[]){

Byte b1=123;
Byte b2=123;
System.out.println(b1==b2);

byte b=123;
Byte b3=new Byte(b);
Byte b4=new Byte(b);
System.out.println(b3==b4);
}
}
```

Lab606.java

```
class Lab606{
public static void main(String as[]){

Boolean b1=true;
Boolean b2=true;
System.out.println(b1==b2);

boolean b=true;
Boolean b3=new Boolean(b);
Boolean b4=new Boolean(b);
System.out.println(b3==b4);
}
}
```

Lab607.java

```
class Lab607{
public static void main(String as[]){

Integer i1=127;
Integer i2=127;
System.out.println(i1==i2);

Integer i3=128;
Integer i4=128;
System.out.println(i3==i4);
}
}
```

Lab608.java

```
class Lab608{
public static void main(String as[]){

Character ch1='A';
Character ch2='A';
System.out.println(ch1==ch2);

Character ch3=171;
Character ch4=171;
System.out.println(ch3==ch4);
}
}
```

Lab609.java

```
class Lab609{
public static void main(String as[]){

Float f1=12.34F;
Float f2=12.34F;
System.out.println(f1==f2);

Double d1=12.34D;
Double d2=12.34D;
System.out.println(d1==d2);
}
}
```

Lab610.java

```
class Lab610{
public static void main(String as[]){

Long x1=123L;
Long x2=123L;
System.out.println(x1==x2);

Long x3=128L;
Long x4=128L;
System.out.println(x3==x4);
}
}
```

Lab611.java

```
class Lab611{
public static void main(String as[]){

Integer x1=250;
Integer x2=250;
System.out.println(x1==x2);

Long x3=250L;
Long x4=250L;
System.out.println(x3==x4);

}
}
Java -Djava.lang.IntegerintegerCache.high=550
Lab611
```

Lab612.java

```
class Lab612{
public static void main(String as[]){

Integer i1=123;
Integer i2=124;
System.out.println(i1==i2);

i1++;

System.out.println(i1==i2);
Integer i3=i1+i2;
System.out.println(i3);
}
}
```

Lab613.java

```
class Hello{
void show(int a,int b){//1
System.out.println("1.show(int,int)");
}
void show(byte a,byte b){//2
System.out.println("2.show(byte,byte)");
}
void show(Byte a,Byte b){//3
System.out.println("3.show(Byte,Byte)");
}
void show(byte... a){//4
System.out.println("4.show(byte...)");
}
}
class Lab522{
public static void main(String as[]){
Hello h=new Hello();
byte b1=10;
byte b2=20;
h.show();
h.show(b1);
h.show(b1,b2);
}
}
```

Lab614.java

```
class Hello{
void show(int a,int b){//1
System.out.println("1.show(int,int)");
}
void show(Byte a,Byte b){//3
System.out.println("3.show(Byte,Byte)");
}
void show(byte... a){//4
System.out.println("4.show(byte...)");
}
}
class Lab523{
public static void main(String as[]){
Hello h=new Hello();
byte b1=10;
byte b2=20;
h.show(b1,b2);

}
}
```

Lab615.java

```
class Hello{
void show(Byte a,Byte b){//3
System.out.println("3.show(Byte,Byte)");
}
void show(byte... a){//4
System.out.println("4.show(byte...)");
}
}
class Lab524{
public static void main(String as[]){
Hello h=new Hello();
byte b1=10;
byte b2=20;
h.show(b1,b2);
}
}
```

Lab616.java

```
class Hello{
void show(byte... a){//4
System.out.println("4.show(byte...)");
}
}
class Lab525{
public static void main(String as[]){
Hello h=new Hello();
byte b1=10;
byte b2=20;
h.show(b1,b2);
}
}
```

SUMMARY

Wrapper Classes:

1) All the Wrapper objects are immutable objects.
2) equals () is overridden in all the wrapper classes to compare the content.
3) hashCode() is overridden in all the wrapper classes.
4) hashCode() of Character class returns the ASCII of the character.
5) hashCode() of Boolean class returns:
    a. 1231 if the value is true.
    b. 1237 if the value is false.
6) hashCode() method of Byte,Short,Integer and Long classes returns the content of the object.
7) All the Wrapper classes are Comparable.

Auto Boxing

8) Compiler uses valueOf() method to convert primitive to wrapper.
9) Compiler uses intValue(), doubleValue() etc to convert wrapper to primitive.
10) When you create Boolean or Byte objects with same content then multiple references point to the same object.
11) When you create Short, Character, Integer and Long object then:

a. When the content is within the range of byte then multiple references point to the same object.
b. When the content is not within the range of byte then multiple references point to the different objects.

12) When you create Float and Double object then always a new object will be created.

13) You can specify the cache range explicitly for Integer as follows:

```
-Djava.lang.Integer.IntegerCache.high=596
```

14) Consider the following case:

```
Object obj =123;          //VALID
        Integer in = new Integer(123); // AUTO-BOXING
        Object obj = in;                    // UP CASTING
```

15) Consider the following case:

```
Long val =123; //INVALID
        Integer in = new Integer(123); // AUTO-BOXING
        Long val = in;                              // UP CASTING
```

Note: You can not assign Integer object to Long variable because of no inheritance relationship between Integer and Long class.

16) In the case of overloaded method calls following checks will happen:

a. First compiler will search for the method with same parameter type.
b. If method with the same parameter type is not found then compiler will search for the method with the wider type parameter.
c. If method with the wider type parameter is also not found then compiler will search for method with Wrapper type parameter.
d. If method with Wrapper type parameter is also not found then only compiler will search for method with Var-Args parameter.

4.2.10. System Class

- System is a final class available in java.lang package and it is used to access the information about your current system.
- System class default constructor is private, so you cannot create the object of System class.
- Members of System class are static, so can be accessed with class name directly.
- Following are the three constants defined in System class:
    - public final static InputStream in;
    - public final static PrintStream out;
    - public final static PrintStream err;
- in is the reference variable of InputStream class and it is pointing to default input device called Keyboard.
- out and err are the reference variable of PrintStream class and these are pointing to default output device called console.
- err will be mainly used to print error message by convention.
- These streams are initialized by the JVM automatically at JVM startup with default INPUT and OUTPUT devices.
- If you have requirement to change default input and output devices to required I/O devies then you can use the following methods:
    - public static void setIn(InputStream);
    - public static void setOut(PrintStream);
    - public static void setErr(PrintStream);

Method	Description
long currentTimeMillis()	Get the current system time as millisecond
void arraycopy(object ar1, int idx1, object ar2, int idx2, int eles)	To copy the array data quickly
Properties getProperties()	Access all the system properties
void setProperties(Properties)	Set the System Properties from Properties object
String getProperty(String pname)	To access the System property for specific Property name
String setProperty(String,String)	Set the System Properties from Properties object
String getenv(String)	Get the ENVIRONMENT VARIABLE value (Implementation from JAVA 5)
void exit(int status)	To Terminate the JVM Programmatically 0 is Normal Termination

	Non-zero is Abnormal Termination
void gc()	To invoke the Garbage-Collector
void loadLibrary(String)	To Load the Native Implementation Libraries

Exploring System.out.println()

System	A final class in java.lang package
out	A static member field of System class and is of type java.io.PrintStream. It is initialized during startup and gets mapped with standard output (CONSOLE) of the machine.
println	A method of java.io.PrintStream. It prints the argument passed and then a new line to the output represented by out field. There are multiple println methods with different arguments (overloading).

Lab617.java

```
class Lab617{
public static void main(String as[]){

System.out.println("I am regular Message");
System.exit(0);
System.err.println("I am Error Message");
System.err.println(System.currentTimeMillis())
;
System.err.println(System.nanoTime());

  System.gc();
  System.runFinalization();
}
}
```

Lab618.java

```
import java.io.*;

class Lab614{
public static void main(String as[]){

PrintStream ps= System.out;

ps.println("I am regular Message");
ps.println("I am Error Message");
ps.println(System.currentTimeMillis());
ps.println(System.nanoTime());

  System.gc();
  System.runFinalization();
}
}
```

Lab619.java

```
import java.io.*;
import java.util.*;

class Lab526{
public static void main(String as[]){

Properties props=System.getProperties();
System.out.println(props);
System.out.println("--------------------");

System.setProperty("my.web.site","myjlc123.com");
System.setProperty("trainer.name","Srinivas Dande");

Enumeration enms=props.propertyNames();
while(enms.hasMoreElements()){
System.out.println(enms.nextElement());
}

System.out.println("--------------------");
System.out.println(System.getProperty("java.io.tmpdir"));
System.out.println(System.getProperty("java.vm.vendor"));
System.out.println(System.getProperty("java.library.path"));
System.out.println(System.getProperty("trainer.name"));
```

4.2.11. Runtime Class

- It is the class in java.lang package and used to interact with the current running JVM or to get the information about the JVM.
- Every java program has the single instance of this class to allow the application to interact with the environment.
- Basically this class is used for memory management and executing additional processes.
- You can define the reference for the Runtime class but you can not create the object.
- The methods of the runtime class are defined as instance, so you need to use some object to invoke the methods.
- Runtime class is implemented using singleton design pattern.
- Singleton class means only one object of that class will be available per JVM.

Method	Description
static Runtime getRuntime()	To get the object of the runtime class
Process exec(String)	To execute any executable file
int availableProcessors()	To get the available processor (Value is NUMBER_OF_PROCESSORS environment variable)
long freeMemory()	Determines the amount of free memory available to JVM (in terms of byte)
long totalMemory()	Determines the amount of total memory used by JVM (in terms of byte)

long maxMemory()	Determines the amount of maximum memory will be used by JVM (in terms of byte)
void exit(int status)	To Terminate the JVM Programmatically 0 is Normal Termination Non-zero is Abnormal Termination
void gc()	To invoke the Garbage-Collector
void loadLibrary(String)	To Load the Native Implementation Libraries

Lab620.java

```
class Lab620{
public static void main(String as[]){

Runtime  rt=Runtime.getRuntime();

System.out.println(rt.availableProcessors());
System.out.println(rt.maxMemory());
System.out.println(rt.totalMemory());
System.out.println(rt.freeMemory());
}
}
```

Specifying Required memory for JVM:

1) From Command Prompt

java -Xmx1024m -Xms512m Lab620

2) In Eclipse

- Go to run configuration
- Select argument tabs
- In VM arguments provide the following

-Xmx1024m -Xms512m

Lab621.java

```
class Lab621{
public static void main(String as[]) throws
Exception{

if(as.length==1){
Runtime  rt=Runtime.getRuntime();
String app=as[0]+".exe";
Process p=rt.exec(app);
}else{
System.out.println("Specify the App name");
}
//Process p1=rt.exec("notepad.exe");
//Process p2=rt.exec("calc.exe");
}
}

/*
Run as follows
java Lab621 calc
java Lab621 notepad
*/
```

4.2.12. Math Class

- It is a final class in java.lang package and contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
- The constructor of the Math class is private, so you can not create the object of Math class.
- All the members of the Math class are static, so you can access members of Math using class name.

Lab622.java

```
public class Lab529 {
public static void main(String[] args) {
System.out.println(Math.E);
System.out.println(Math.PI);
System.out.println(Math.sqrt(16));
System.out.println(Math.pow(3, 2));
System.out.println(Math.abs(-99));
System.out.println(Math.max(20, 10));

System.out.println(Math.min(20, 10));
System.out.println(Math.max(99.9, 20.5));
System.out.println(Math.min(99.9, 20.5));
System.out.println(Math.ceil(10.2));
System.out.println(Math.ceil(10.5));
System.out.println(Math.ceil(10.9));
System.out.println(Math.floor(10.2));
System.out.println(Math.floor(10.5));
System.out.println(Math.floor(10.9));
System.out.println(Math.round(10.2));
System.out.println(Math.round(10.5));
System.out.println(Math.round(10.9));
}
}
```

Lab623.java

```
public class Lab530 {
public static void main(String[] args) {
for (int i = 0; i < 10; i++) {
System.out.println(Math.random());
}

System.out.println("\n*****");
for (int i = 0; i < 10; i++) {
System.out.println(10 * Math.random());
}
}
}
```

4.3 java.math Package

4.3.1 BigInteger Class

- BigInteger class is a class available in java.math package.
- This class is subclass of java.lang.Number class.
- When you have a requirement to use the value out side the range of long then you can use this class.
- It allocates the memory as per the values available in the Object.

Lab624.java

```
import java.math.BigInteger;
public class Lab534 {
public static void main(String[] args) {
BigInteger bint1 = new BigInteger("4");
System.out.println(bint1.bitCount());
        //Number  of 1 bit
System.out.println(bint1.bitLength());
        // Number of total bits Allocated
long val = 92233720368547758O7L;   // MAX
VALUE
long val2 = 100;
long res = val + val2;
System.out.println(res);
} }
```

Lab625.java

```
import java.math.BigInteger;
public class Lab535 {
public static void main(String[] args) {
BigInteger in1 = new
BigInteger("9223372036854775807");
BigInteger in2 = new BigInteger("100");
BigInteger res1 = in1.add(in2);
System.out.println(res1);
}
}
```

4.3.2 BigDecimal Class

- BigDecimal class a class available in java.math package.
- This class is subclass of java.lang.Number class.
- When you have a requirement to use the value out side the range of double then you can use this class.
- It allocates the memory as per the values available in the Object.

Lab626.java

```
import java.math.BigDecimal;
public class Lab536 {
public static void main(String[] args) {
double d1 = Double.MAX_VALUE;
double res = d1 + 100;
System.out.println(d1);
System.out.println(res);
System.out.println("\n***");
BigDecimal dec1 = new
BigDecimal(Double.MAX_VALUE);
System.out.println(dec1);
BigDecimal dec2 = new BigDecimal(100);
BigDecimal res2 = dec1.add(dec2);
System.out.println(res2);
BigDecimal dec1=new
BigDecimal(Double.POSITIVE_INFINITY);
System.out.println(dec1);
}
}
```

Assignment # 17

Q1) What is Wrapper class and How many Wrapper classes are available?

Q2) What is the way to use for following conversions:

a. Primitive to String
b. String to Primitive
c. Primitive to Object
d. Object to Primitive
e. Wrapper to String
f. String to Wrapper
g. Integer to Binary
h. Integer to Hexadecimal
i. Integer to Octal

Q3) What is Boxing and Un-Boxing?

Q4) Write the inheritance hierarchy for Wrapper classes?

Q5) What is Auto-boxing and Auto-unboxing?

Q6) What is cashing concept with Wrapper classes?

Q7) What is System class?

Q8) Why cannot I instantiate System class?

Q9) How to modify the default I/O streams to required I/O streams?

Q10) Explore System.out.println()?

Q11) What is Runtime class and what information I can get through Runtime class methods?

Q12) How to instantiate Runtime class?

Q13) What is the use of Math class?

Q14) What is the use of BigInteger and BigDecimal class and in which package these classes are available?

Q15) What is Singleton Design Pattern? Write the steps to write custom Singleton class?

Q16) What is marker interface? Write the steps to write custom marker interface?

Q17) What is immutable class? Write the steps to write custom immutable class?

Practice Test # 17

Q.No	Question	Options	Answer

1	public class Test {
	public static void main(String[] args) {
	System.out.println(Byte.MIN_VALUE);
	System.out.println(Byte.MAX_VALUE);
	System.out.println(Byte.SIZE);
	}
	}
		A) -128 127 8
		B) -127 128 8
		C) Runtime Error
		D) Compilation Error	

2	public class Test {
	public static void main(String[] args) {
	byte b1 = 10;
	Byte b = new Byte(b1);
	System.out.println(b.hashCode());
	}
	}
		A) 10
		B) 584521
		C) Runtime Error
		D) Compilation Error	

3	public class Test {
	public static void main(String[] args) {
	String str = Long.toBinaryString(5L);
	System.out.println(str);
	System.out.println(Long.toHexString(15L));
	System.out.println(Long.toOctalString(8L));
	}
	}
		A) 101 f 12
		B) 101 f 10
		C) 5 f 12
		D) Runtime Error	

4	public class Test {
	public static void main(String[] args) {
	Float f1 = new Float(0.0f / 0.0f);
	System.out.println(f1.isInfinite());
	System.out.println(f1.isNaN());
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

5	public class Test {
	public static void main(String[] args) {
	Float f1 = new Float(10.0f / 0.0f);
	System.out.println(f1.isInfinite());
	System.out.println(f1.isNaN());
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

6	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isDigit('9'));
	System.out.println(Character.isDigit('A'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

7	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isLetter('A'));
	System.out.println(Character.isLetter('9'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

8	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isLetterOrDigit('A'));
	System.out.println(Character.isLetterOrDigit('9'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

9	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isMirrored(']'));
	System.out.println(Character.isMirrored('H'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

10	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isJavaIdentifierStart('8'));
	System.out.println(Character.isJavaIdentifierStart('$'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

11	public class Test {
	public static void main(String[] args) {
	System.out.println(Character.isJavaIdentifierPart('8'));
	System.out.println(Character.isJavaIdentifierPart('$'));
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

12	public class Test {
	public static void main(String[] args) {
	Integer in1 = 127;
	Integer in2 = 127;
	Integer in3 = new Integer(127);
	System.out.println(in1 == in2);
	System.out.println(in1 == in3);
	}
	}
		A) true false
		B) false true
		C) true true
		D) false false	

13	public class Test {
	public static void main(String[] args) {
	Hello h = new Hello();
	h.show('A');
	}
	}
	class Hello{
	void show(char... values){
	System.out.println("CHAR1");
	}
	void show(int ab){
	System.out.println("INT2");
	}
	}
		A) INT2
		B) CHAR1
		C) Runtime Error
		D) Compilation Error	

14	public class Test {
	public static void main(String[] args) {
	String os=System.getProperty("os.name");
	System.out.println(os);
	}
	}
		A) Windows XP
		B) Windows 7
		C) LINUX
		D) Depends at Runtime	

15	public class Test {
	public static void main(String[] args) {
	int x = new Integer(10);
	x++;
	System.out.println(x);
	}
	}
		A) 10
		B) 11
		C) Runtime Error
		D) Compilation Error	

16	import java.math.BigInteger;
	public class Test {
	public static void main(String[] args) {
	BigInteger in1 = new BigInteger("2147483648");
	BigInteger in2 = new BigInteger("1");
	System.out.println(in1.add(in2));
	}
	}
		A) 2147483649
		B) 2147483648
		C) Runtime Error
		D) Compilation Error	