Internationalization (I18N)

I18N :-

→ various Countries follow various Conversions to represent dates & no.'s e.t.c

→ Our application should generate Locale Specific responses like for India people the response should be interms of Rs. (Rupees) & for the U.S people the response should be interms of dollars(₹) the process of designing such type of web application is called "Internationalization" (I18N).

→ We can implement I18N by using the following classes

① Locale
② NumberFormat
③ DateFormat

Locale :-

→ A Locale Object represents a Geo-graphic Location

Constructors :-

→ We can Create a Locale object by using the following Constructor.

(1) Locale l = new Locale(String language);

(2) Locale l = new Locale(String language, String Country);

javap javautil Locale.

→ Locale class defines Several Constants to represent Some Standard Locales. We can use these Locale directly without Creating on our own.

Eg:- Locale.US
Locale.ITALIAN
Locale.ENGLISH
Locale.UK

Note:-

→ Locale class is the final class present in java.util package

→ It is the direct child class of Object it implements Cloneable & Serializable interfaces.

Important methods of Locale class:-

① public static Locale getDefault();

② public static void setDefault(Locale l);

③ public String getLanguage(); en

④ public String getDisplayLanguage(); english

⑤ public String getCountry(); us

⑥ public String getDisplayCountry(); unitedstates

⑦ public static String[] getISOCountries();

⑧ public static String[] getISOLanguages();

⑨ public static Locale[] getAvailableLocales();

Ex:-

```
import java.util.*;
Class LocaleDemo1
{
 Public static void main(String[] args)
 {
    Locale l1 = Locale.getDefault();
    S.o.pln(l1.getCountry() + "...." + l1.getLanguage());
    S.o.pln(l1.getDisplayCountry() + "----" + l1.getDisplayLanguage());
    Locale l2 = new Locale("pa","IN");
    Locale.setDefault(l2);
    String[] s3 = Locale.getISOLanguages();
    for(String s4 : s3)
    {
      S.o.pln(s4);
    }
    String[] s4 = Locale.getISOCountries();
     for(String s5 : s4)
     {
       S.o.pln(s5);
     }
     Locale[] s = Locale.getAvailableLocales();
     for(Locale s1:s)
     {
       S.o.pln(s1.getDisplayCountry() + "---" + s1.getDisplayLanguage());
     }
};
```

NumberFormatt Classes :-

→ Various Countries follow various Conversions to represent Number by using NumberFormatt Class we Can formatt a numb according to a particular Locale.

→ NumberFormatt class present in java.text package & it is an abstract class. Hence we Can't Create NumberFormatt object directly

✗ NumberFormatt nf = new NumberFormatt();

Creating NumberFormatt object for the default Locale :-

→ NumberFormatt Class defines the following methods for this

① public static NumberFormatt getInstance();

② public static NumberFormat getCurrencyInstance();

③ public static NumberFormat getPercentInstance();

④ public static NumberFormat getNumberInstance();

Getting NumberFormat object for a Specific Locale :-

→ we have to pass the Corresponding Locale object as argument to the above methods

Ex:-

① public static NumberFormat getCurrencyInstance(Locale l);

→ Once we got NumberFormat Object we can format & parse numbers by using the following methods of NumberFormat class

① public String format(long l);

② public String format(double d);

→ To format (or) Convert java specific number form to Locale Specific String form.

③ public Number parse(String s) throws ParseException

→ To Convert Locale Specific String form to java Specific Number form.

Ex1:-

w.a.p to represent a java number in Italy Specific form.

```
import java.text.*;
import java.util.*;
class NumberFormatDemo2
{
  P.S.V.m(___)
  {
    double d=123456.789;
    NumberFormat nf=NumberFormat.getInstance(Locale.ITALY);
    S.o.pln("Italy form is:"+nf.format(d));
  }
}
```

o/p:- Italy form is: 123.456,789

Eg:- w.a.p to represent a java number in India, U.K & U.S Currency forms.

```
import java.text.*;
import java.util.*;
Class NumberFormatDemo3
{
   P.S.V.m(------)
   {
      double d = 123456.789;
      Locale India = new Locale("pa","IN");   // any Location in India
      NumberFoamat nf1 = NumberFoamat.getCurrencyInstance(India);
      S.o.p1p("India Notation is...." + nf1.format(d));
      NumberFoamat nf2 = NumberFoamat.getCurrencyInstance(Locale.US);
      S.o.pln("US Notation is...." + nf2.format(d));
      NumberFoamat nf3 = NumberFoamat.getCurrencyInstance(Locale.UK);
      S.o.pln("UK notation is...." + nf3.format(d));
   }
}
```

O/P:- India Notation is ..... INR 123,456.79
US Notation is $123,456.79
Uk Notation is ..... Ù 123,456.79

Setting Maximum & minimum integer & fraction digits:-

→ NumberFormat class defines the following methods to set Maximum & minimum fraction & integer digits.

① public void setMaximumFractionDigits(int n);

② public void setMinimumFractionDigits(int n);

③ public void setMaximumIntegerDigits(int n);

④ public void setMinimumIntegerDigits(int n);

18/5/11

Ex1.

```
NF nf = NF.getInstance();

① nf.setMaximumFractionDigits(2);
    S.o.pln(nf.format(123.4567)); // 123.45
    S.o.pln(nf.format(123.4)); //123.4

② nf.setMinimumFractionDigits(2);
    S.o.pln(nf.format(123.4567)); // 123.4567
    S.o.pln(nf.format(123.4)); // 123.40

③ nf.setMaximumIntegerDigits(3);
    S.o.pln(nf.format(123456.234)); // 456.234
    S.o.pln(nf.format(12.3456)); // 12.3456

④ nf.setMinimumIntegerDigits(3);
    S.o.pln(nf.formatt(123456.234)); //456.234
    S.o.pln(nf.formatt(12.3456)); // 012.3456
```

DateFormat class :-

→ Various Countries follow various Conventions to represent Date.

→ By using DateFormatt class we can formatt the DATE according to a particular Locale.

→ DateFormat class is an abstract class & present in java.text pack.

Getting DateFormat object for Default Locale :-

DateFormat class defines the following methods for this

(1) public static Dateformat getInstance();

(2) public static DateFormat getDateInstance();

(3) public static Dateformat getDateInstance(int Style);

→ DateFormat.Full → 0
DateFormat.LONG → 1
DateFormat.MEDIUM → 2
DateFormat.SHORT → 3

Getting Dateformat object for the Specific Locale :-

① public static DateFormat getDateInstance(int Style, Locale l);

→ Once we got DateFormat Object we can format & paase dates by using the following methods.

① public String format(Date d);

→ To Convert Java Date form to Locale Specific String form

Note:-
Default Style is Medium & most of the cases default Locale is US

② Public Date parse(String s) throws ParseException

To Convert Locale Specific Date Form to java DateForm.

Ex:-

W.a.p To display System Date in all possible Styles of U.S format

```
import java.util.*;
import java.text.*;
Class DateFormatDemo1
{
  p.s.v.m(——)
  {
    S.o.pln("full form:" + DateFormat.getDateInstance(0).
                                    format(new Date()));

          (or)

  // DateFormat df = DateFormat.getDateInstance(0);
     S.o.pln(df.format(new Date());

  S.o.pln("Long form:" + DF.getDateInstance(1).format(new Date));
  S.o.pln("MEDIUM form:" + DF.getDateInstance(2).Format(new Date()));
  S.o.pln("SHORT form:" + DF.getDateInstance(3).format(new Date()));
  }
}
```

O/P:- full form: Thursday, february. 2. 2010

Long form : February 18, 2010

Medium form : Feb 18, 2010

Short form : 2/18/10

Ex 2):

(P) w.a.p to display System Date US, UK & Italy form.

```
{
    S.o.pln("US form:" + DF.getDateInstance(0, Locale.US).format(
                                                    new Date());
    S.o.pln("UK form:" + DF.getDateInstance(0, Locale.UK).format(new Date()));
    S.o.pln("ITALY form:" + DF.getDateInstance(0, Locale.ITALY).format(new Date()));
}
```

o/p:

US form : Tuesday, May 18, 2010

UK form : Tuesday, 18 May 2010

ITALY form: martedao 18 maggie 2010

Getting Date-format object to represent both DATE & TIME :-

① public static DateFormat getDateTimeInstance();

② public static DateFormat getDateTimeInstance(int dateStyle, int timeStyle);

③ public static DateFormat getDateTimeInstance(int dateStyle, int timeStyle, Locale l);

Ex:

```
S.o.pln("US form:" + DateFormat.getDateTimeInstance(0, 0, Locale.US)
                                                    .format(new Date()));
```

o/p:

US form : Tuesday, May 18, 2011 9:53:45 AM GMT:+5:30

Note:- Default style is medium & most of the cases default Locale is OS

Development

javac :-

We can use this Command to Compile a Single or group of .java files.

Syn:- javac [options] A.java/

A.java B.java

*.java

[options]
- -d
- -Source
- -cp
- -classpath
- -version

java :-

We can use java Command to run a .class file

Syn:- java [options] A

[options]
- -ea | -esa | -da | -dsa
- -version
- -cp / -classpath
- -D
- :

Note:- We can Compile a group of .java files at a time where as we can run only on .class file at a time.

Classpath :-

→ Classpath describes the location where required .class files are available.

→ JVM will always use Classpath to locate the required .class file.

→ The following are various possible ways to set the Classpath.

① Permanently by using Environment variable classpath.

→ This classpath will be preserved after system restart also

② At Command prompt level by using Set Command.

Set classpath = %classpath%;D:\path>

→ This classpath will be applicable only for that particular Comand prompt window only. Once we close that Command prompt automatically classpath will be lost

③ At Command Level by using -cp option

java -cp D:\path> Test ↵

→ This classpath is applicable only for this particular Command. Once Command execution Completes automatically classpath will be lost.

* Among the above 3 ways the most Commonly used approach is Setting classpath at Command Level.

Ex:-
```
class Test
{
  p.s.v.m(——)
  {
    S.o.pln("Classpath Demo");
  }
}
```

D:\Durgaclasses\> javac Test.java ↵

>java Test ↵

O/P:- Classpath Demo

✗ D:\java Test ↵ R.E:- NoClassDeffoundError

✓ D:\> java -cp D:\Durgaclasses Test ↵ ✓

O/P:- ClasspathDemo

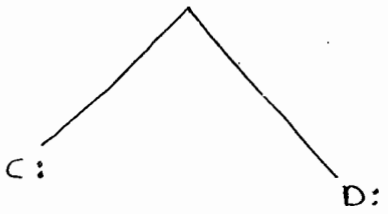
✓ G:\> java -cp D:\Durgaclasses Test ↵

O/P:- ClasspathDemo

Note:-

If we set classpath explicitly then we can run Java program from any location but if we are not setting the classpath then we have to run java program only from current working directory.

Ex:-




```
public class fresher
{
  public void m1()
  {
    S.o.pln ("I want JOB");
  }
}
```

```
class Company
{
  p.s.v.m (-->)
  {
    fresher f = new fresher();
    f.m1()
    S.o.pln("Getting JOB is very
             easy.. Not required to
             crazy");
  }
}
```

C:\> javac fresher.java ✓

D:\> javac Company.java ✗

C.E:- Cannot find Symbol
Symbol: class fresher
location: class Company

D:\> javac -cp C: Company.java ✓

✗ D:\java Company ←

R.E:- NoClassDefFoundError: fresher

✗ D:\ java -cp C: Company ←

R.E:- NoClassDefFoundError: Company

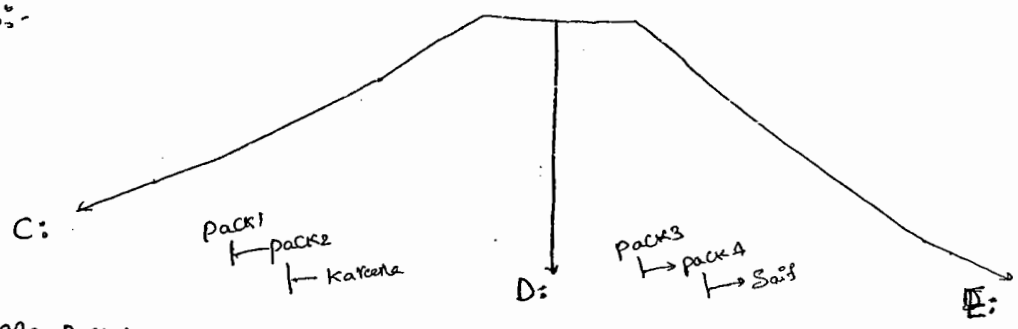
✓ D:\ java -cp D:;C: Company (or) D:\java -cp .;C: Company

o/p :- I can JOB

Getting JOB is very easy... not required to worry.

✓ E:\ java -cp D:;C: Company

Ex3:-




```
Package pack1.pack2;
Public class Kareena
{
 public void m1()
 {
S.o.pln(" Hello Saif Can U please see hello func");
 }
}
```

```
package pack3.pack4;
import pack1.pack2.kareena
Public class Saif
{
 public void m1()
 {
 Kareena k = new kareena();
 k.m1();
 S.o.pln(" Not possible.. As I am in SUP class");
 }
}
```

```
import pack3.pack4.Saif;
class Durga
{
 p.s.v.m(--)
 {
 Saif s = new Saif();
 s.m1();
 S.o.pln(" Can I help U");
 }
}
```

✓ C:\> java -d. Kareena

✓ D:\> java -d. Saif.java

C.E:- Cannot find Symbol

Symbol : class Kareena

Location : class Saif

✓ D:\java -cp c: -d . Saif.java

✗ E:\javac Durga.java

C.E:- Cannot find Symbol
Symbol: class Saif
location: class Durga

✓ E:\>javac -cp D: Durga.java

✗ E:\>java Durga ↵

R.E:- NoClassDefFoundError : Saif

✗ E:\>java -cp D: Durga ↵

R.E:- NoClassDefFoundError : Durga

✗ E:\> java -cp .;D: Durga

R.E:- NoClassDefFoundError : Bisreena

✓ E:\>java -cp E:;D:;C: Durga

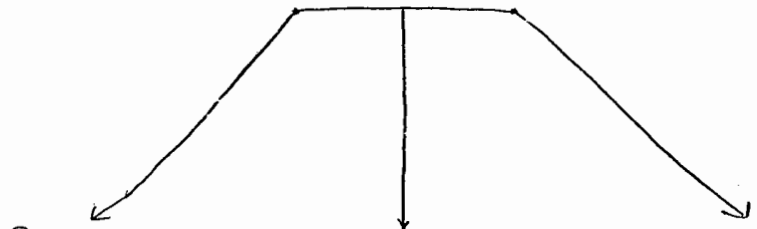
Note:-

① Compiler will check only one level dependency where as JVM will check all levels of dependency

② If any folder structure created because of package statement it should be resolved through import statement only. & Base package location we have to update in classpath.

③ within the classpath the order of locations is very important for the required .class file, JVM will always search the locations from

Left → Right in classpath. Once JVM finds the required file then the rest of the classpath won't be searched.



C:

```
Class Nagavali
{
 p.s.v.m(--)
 {
 S.o.pln("C: Nagavali");
 }
}
```

D:

```
Class Nagavali
{
 p.s.v.m(--)
 {
 S.o.pln("D: Nagavali");
 }
}
```

E:

```
Class Nagavali
{
 p.s.v.m()
 {
 S.o.pln("E: Nagavali
 }
}
```

```
C:\> javac Nagavali.java
D:\> javac Nagavali.java
E:\> javac Nagavali.java
C:\> java Nagavali
```

o/p C: Nagavali

```
D:\> java -cp C:;D:;E: Nagavali
```

o/p C: Nagavali

```
D:\> java -cp E:;D:;C: Nagavali
```

o/p: E: Nagavali

```
D:\> java -cp D:;E:;C: Nagavali
```

o/p: D: Nagavali

JAR file :-

→ If Several dependent files are available then it is never recommended to set each class file individually in the classpath we have to group all those .Class file into a single Zip file & we have to make that Zip file available in the class path. This zip file is nothing but JAR file.

Ex(1) :-

To develope Servlet all required .class files are available in Servlet-api.jar. we have to make this Jar file available in the classpath then only Servlet will be Compiled.

jar vs war vs ear :-

① jar : (java archieve file)

→ It Contains a group of .class files

② war :- (web archieve file)

→ It represents a web application which may Contain Servlets, Jsps, HTMLs, CSS file, JavaScripts, e.t.c..

③ ear :- (Enterprise archeive file)

→ It represents an enterprise application which may Contains Servlets, Jsps, EJBs, JNS Components e.t.c.

Various Commands :-

① To Create a jar file :-

```
jar -cvf durga.jar A.class B.class C.class
                   *.class
```

② To extract a jar file :-

```
jar -xvf durga.jar
```

③ To Display table of contents of a jar file :-

```
jar -tvf durga.jar
```

Ex :-

```
public class DurgaColorfullCalc
{
 public static int add(int x, int y)
 {
  return x*y;
 }
public static int add(int x, int y)
 {
  return 2*x*y;
 }
}
```

```
C:\> javac DurgaColorfullCalc.java
C:\> jar -cvf durgaCalc.jar DurgaColorfullCalc.class
```

```
class Bakasra
{
 p.s.v.m(——)
 {
 s.o.pln(DurgaColorfulCalc.add(10,20));
 s.o.pln(DurgaColorfulCalc.multply(10,20));
 }
}
```

✗ D:\> javac Bakasra.java

✗ D:\> javac -cp c: Bakasra.java

✓ D:\> javac -cp c:\durgaCalc.jar Bakasra.java

✓ D:\> javac -cp .;c:\durgaCalc.jar Bakasra.

o/p:- 200
400

Note:-

→ when ever we are placing a jar file in the classpath Compulsary name of the jar file we should include, just Location is not enough.

Shoart cut way to place jar file :-

→ If we are placing the jar file in the following Location then it is not required to set classpath explecitly bydefault it is available to Jvm & Java Compiler.

JDK
 ⟶ JRE
  ⟶ Lib
   ⟶ ext
    ⟶ *.jar

System properties :-

→ For every System persistance information will be maintain in the form of System properties. These may include O.S name, Varchual machine version, User Country .e.t.c....

→ We can get System properties by using getProperties() method of System class

Ex!- Demo program to print all System properties.

```
import java.util.*;
class Test
{
  public static void main(String[] args)
  {
    Properties p = System.getProperties();
    p.list(System.out);
  }
}
```

→ We can Set System property from the Command prompt by using -D option

Space is not allowed

ex!- Java -Ddurga=SCJP Test

durga → name of the property

SCJP → Value of the property

9) JDK Vs JRE Vs JVM :-

JDK:(Java developement kit) :-

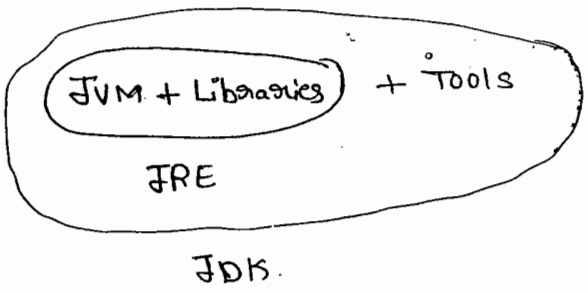
→ To develop & run Java application the required environment provided by JDK.

JRE:- (Java Runtime Environment) :-

→ To run Java application the required environment provided by JRE

JVM :-

→ This machine is responsible to execute Java program.



JDK = JRE + Tools

JRE = JVM + Libraries.

Note:-

→ On client machine we have to install JRE, where as on the developers machine we have to install JDK.

diff. b/w path & classpath :-

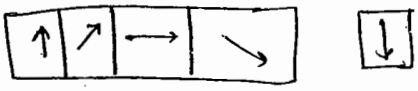
→ We can use classpath to describe the location where required .class files are available.

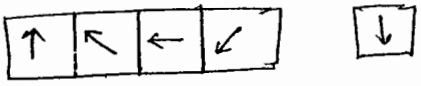
→ If we are not setting the classpath then our program won't be run.

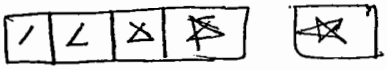
Path :-

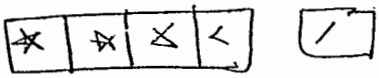
→ We can use path variable to describe the Location where required Binary executables are available.

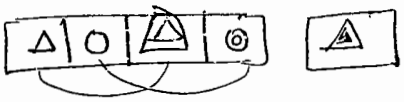
→ If we are not setting path variable then java & javac Commands won't work.

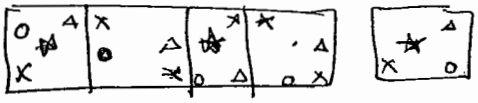
7) Clockwise 

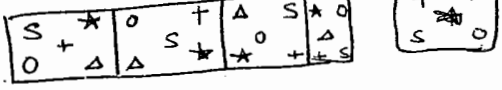
9) Anticlock 

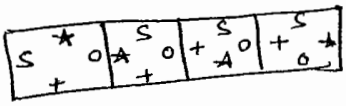
③ Increasing 

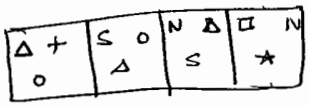
④ Decreasing 

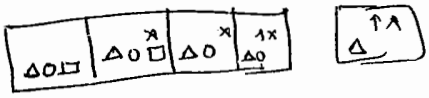
⑤ Alternate 

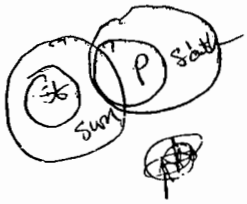
⑥ multiple movement 

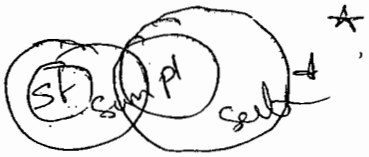
⑦ Rotation 

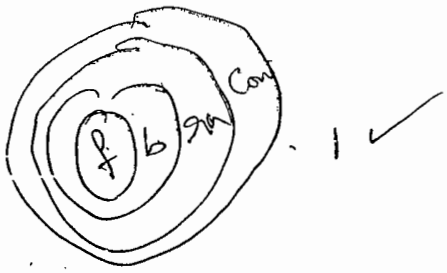
⑧ Interchange 

⑨ missing of fig 

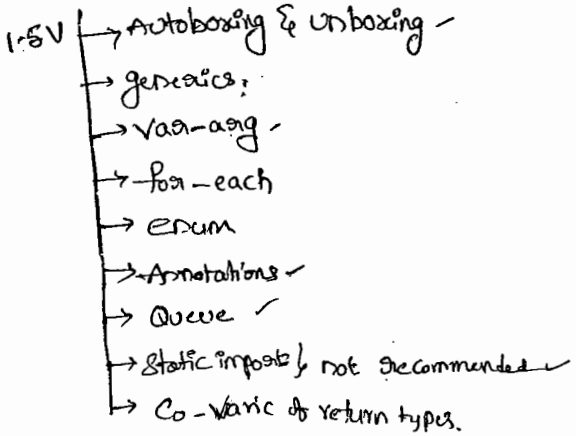
⑩ Substitution 







miracle



Walk
↓
Jogging
↓
Running
↓
Sprinting

Siddhartha (VNR VJIT)
9951884313
Siddharthaprs@yahoo.co.in

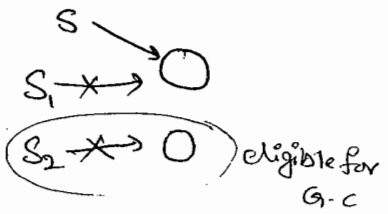
Vishnuteja.Y.S
9703340473, 9493410648
Vishnuteja87@gmail.com

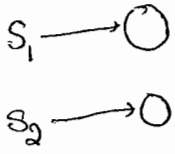
Vasu - 8179969444 (CEVMG)
Siva Sarma @ gmail.com.

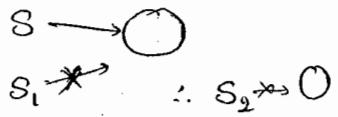
Ex 2:-

```
Class Test
{
 p.s.v.m(String[] args)
 {
  Student s = m1();
 }
 p.s.Student m1()
 {
  Student s1 = new Student();
  Student s2 = new Student();
  return s1;
 }
}
```

one object eligible for G.C →



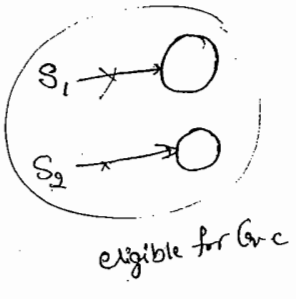




Ex 3 :-

```
Class Test
{
 p.s.v.main(String[] args)
 {
  m1();
 }
 p.s.Student m1()
 {
  Student s1 = new Student();
  Student s2 = new Student();
  return s1;
 }
}
```

Two objects eligible for G.C →

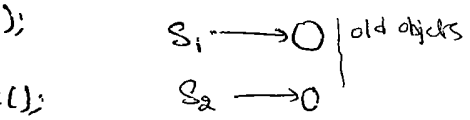


2) Reassigning the Reference variable :-

→ If an object is no longer required then reassign its reference variables to some other objects then that old object automatically eligible for G.C.

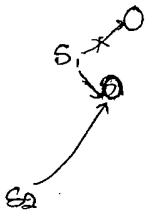
Ex(1) :-

```
Student S1 = new Student();
Student S2 = new Student();
```

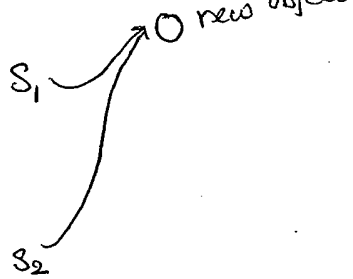


no object eligible for G.C →

```
S1 = new Student();
```



One object eligible for G.C →




```
S2 = S1;
```

Two objects eligible →

(3) Objects Created Inside a method :-

→ The Objects which are Created inside a method are by default eligible for G.C after Completing that method.

Ex :-

```
class Test
{
  p.s.v.main(String [] args)
  {
    m1();
  }
  p.s.v.m1()
  {
    Student S1 = new Student();
    Student S2 = new Student();
  }
}
```

2 objects eligible for G.C →