Internationalization (I18N)

I18N

- Various countries follow various convention to represent dates & no's etc.
- Our application should generate locate specific response like for India people the response should be in term of Rs(rupees) & for the U.S. pepole the response should be in term of dollars($) the process of design such type of web application is called Internationalization(I18N).
- We can implement I18N by using the Following classes.
  1)Local
  2)NumberFormat
  3)DateFormat

1)Local:

A local object represents a Geo-graphic Location.

Constructor:

- We can create a local object by using the following constructor
  1) Locale l=new Locale(String language);
  2) Locale l=new Locale(String language, String Country);
- Local class defines several constants to represent some standard Locales we can use These Locale Directly without creating on own.
  Example:

Locale US	Locale.ENGLISH
Locale.ITALITAN	Locale.UK

Note:

- Locale class is the final present in java.util package
- It is the direct class of object it implements clonable & serializable interface.

Importance method Of Locale Class:

1) public static Locale getDefault();
2) public static Locale getDefault(Locale l);
3) public String getLanguage();
4) public String getDislpayLanguage();
5) public String getCountry();
6) public String getDisplayCountry();
7) public static String[] getISOCountries();
8) public static String[] getISOLanguage();
9) public static Locale[] getAvailableLocale();

Example:-

```
Import java.io.*;
Class LocaleDemo
 {
  Public static void main(String args[])
  {
     Locale l1=locale.getDefault();
     System.out.println("l1.getCountry()+"---"+l1.getLanguage()");
     System.out.println("l1.getDisplayCountry()+"---
"+l1.getDislpayLanguage()");
     Locale l2=new Locale("pa","iN");
     Locale.setDefault(l2);
     String[]s3=Locale.getISOLanguage();
     for(String s4:s3)
      {
        System.out.println(s4);
      }
String[] s4=Locale.getISOCountries();
      for(String s4:s3)
      {
        System.out.println(s5);
      }
      Locale[] s=Locale s=Locale.getAvailableLocales();
```

```
    For(Locale s1,s)
    {
    System.out.println("s1.getDisplayCountry()+"---
"+s1.getDislpayLanguage()");
    }
   }
}
```

Number Format Classes:-

- Various countries follow various convention to represent number by using Number Format classes we can format a number according to a particular Locale.
- Number Format class parent in java.next package & it is an abstract class.hence we can,t create NumberFormat object directly

NumberFormat nf=new Number Format(); //invalid

- **Creating NumberFormat object for the default Locale:-**

  NumberFormat classes defines the following methods for this
  1) Public static NumberFormat getInstance();
  2) Public static NumberFormat getCurrencyInstance();
  3) Public static NumberFormat getParcentInstance();
  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
    1) Public static NumberFormat getCurrencyInstance(Locale l);
  - once we get NumberFormat object we can format & parse numbers by using the Following methods of NumberFormat classes.
    1) public String format(lang l);

2) public String format(double d);

- To format or convert java specific NumberFormat to locale specific String Form.

3) public Number parse(String s)Throws ParseException

- To covert Locale specific string form to java Specific number form

Example 1:

Write a program to represent a java number in Italy Specific form.

```
import java.io.*;
import java.util.*;
class NumberFormatDemo2
 {
   public static void main(String args[])
   {
     Double d=123456.789
  NumberFormat
nf=NumberFormat.getInstance(Locale.ITALY);
   System.out.println("Italy format:"nf.format(d));
}
}
```

Output:

Italy form is:123.456,789

Example 2:

Write a program to represent a java number in India.UK & US currency forms.

```
import java.test.*;
import java.util.*;
class NumberFormatDemo3
 {
 public static void main(String args[])
 {
  Double d=123456.789;
   Locale India=new Locale("pa","IN");
 NumberFormat nf1= NumberFormat.getcurrencyInstance(india);
  System.out.println("India Notation is--------:"+nf1.format(d));
 NumberFormat nf2= NumberFormat.getcurrencyInstance(locale.US);
  System.out.println("US Notation is--------:"+nf2.format(d));
 NumberFormat nf3= NumberFormat.getcurrencyInstance(locale.UK);
  System.out.println("UK Notation is--------:"+nf3.format(d));

}
}
```

Output:

India Notation is--------INR 123.456.79

US Notation is----------$123,456.79

UK Notation is--------U 123,456.79

Setting maximum & Number integers & Fraction digits.

- NumberFormat class defines the following methods to set maxim um & minimum fraction & integer digits.
  1) public void SetMaximumfractionDigit(int n);
  2) public void SetMinimumfractionDigit(int n);
  3) public void SetMaximumIntegerDigit(int n);
  4) public void SetMinimumIntegerDigit(int n);

  Example:

  NF nf=NF.getInstance();

  1) nf.SetMaximumfractionDigit(2);
     System.out.println(nf.format(123.4567)); //123.45
     System.out.println(nf.format(123.4)); //123.4

2) nf. SetMinimumfractionDigit(2);

   System.out.println(nf.format(123.4567)); //123.4567
   System.out.println(nf.format(123.4)); //123.40

3)nf. SetMaximumIntegerDigit(3);

   System.out.println(nf.format(123456.234)); //456.234
   System.out.println(nf.format(12.3456)); //12.345

4)nf. SetMinimumIntegerDigit(3);

   System.out.println(nf.format(123456.234)); //456.234
   System.out.println(nf.format(12.3456)); //012.3456

- **DateFormated class:**
  - Various countries follow various Convensions to represent date
  - By using dateFormat class we can format the DATE according to a particular Locale.
  - DateFormat class is an abstract class & parent in java.Text pair.

- **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.Full	1
DateFormat.Full	2
DateFormat.Full	3

- **Getting DateFormat object for Specific Locale:**

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

Once we get DateFormat object we can format & pass dates by using the following methods.

1) public String format(Date d);
   To convert Java Date form to Locale specific String form.
2) public date parse (String s)throws parseException.

Example:

Write a Program to display System Date in all Possible styles of US format.

```
import.java.util.*;
import.java.test.*;
class DateFormatDemo1
{
  public static void main(String args[])
  {
System.out.println("full form:"+DateFormat.getDateInstance(0).format(new Dates()));

                    (or)

  // DateFormat df=DateFormat.getDateInstance(0);

    System.out.println("df.format(new Date())");

     System.out.println("Long form:"+DF.getDateInstance(1).format(new.Date()));

     System.out.println("MEDIUM form:"+DF.getDateInstance(2).format(new.Date()));

     System.out.println("Long form:"+DF.getDateInstance(3).format(new.Date()));

  }
}
```

OutPut:

Full form:Thursday,February.2.2010
Long form:February 18.2010
Medium form:Feb 18,2010.

Example 2 :

Write a proram to display system date Us,UK & Italy form

{

```
System.out.println("US
form:"+DF.getDateInstance(0,Locale.US).format(new.Date()));

System.out.println("UK
form:"+DF.getDateInstance(0,Locale.UK).format(new.Date()));

System.out.println("ITALY
form:"+DF.getDateInstance(0,Locale.LTALY).format(new.Date()));
```

}

OutPut

US form:Tuesday,may 18,2010
UK form:18 May2010
ITALY form: 18 maggie 2010.

- **Getting DateFormat object to Reprsent 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);

Example

System.out.println("US form:"+DF.getTimeInstance(0,0,Locale.US).format(new.Date()));

Output:

US form:Tuesday,may 18,2010 9:53:45 AM Gmt:+5:30

Note:

Default style is medium & most of the cases default Locale is US.

Default style is Medium & most of the cases default Locale is US.

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:

we can use java Command to a run a .class file.

Syn

java[option] A

Note:- we can compile a group of .java files at a time where as we can run only one .class file at a time.

Class path:

- classpath described the location where required .class file are available.
- JVM will always to locate the required .class file.
- The Following are various possible ways to set the classpath.

1)Permenently by using environment variable classpath

- This class path will be preserved after system restart also.

2)At command prompt level by using set Command.

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

- This class path will be only for that particular command prompt once command execution completes automatically classpath will be lost.

3)At command level by using cp option

java-cpD:\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 or command level.

Example:

```
Class Test
  {
   public static void main(String args[])
   {
    System.out.println("Classpath  Demo");
}
}
```

D:\Durgaclass\>javac Test.java

>Java Test //valid
O/p= Classpath Demo

D:\.java Test //R.E.NoClassDefoundError //invalid

D:\>java-cp D:\Durgaclasses Test //valid

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 directly.

Example 2)

C:	D:

public class fresher
{
public void m1()
{
System.out.println("I Wan't Job");
}
}
	class Company
	{
	Public static void main(string args[])
	{
	fresher f=new fresher();
	f.m1();
	System.out.println("Getting a job is very
	easy not required to warry");
	}
	}

C:\> javac Fresher.java //valid

D:\> javac Company.java // cannot find symbol

Symbol:Class Fresher

Location: class Company

D:\> javac –cp c:Company.java //valid

D:\javac Company //R.E. NoClassDefoundError:Fresher

D:\ java-cp c:Company //R.E. NoClassDefoundError:Company

D:\java-cp D:;C: Company D:\java-cp;C:Company

o/p= I wan't Job

Getting Job is very eassy not required to warry

E:\java-cp D:;C: Company

Example 3)

C:	D:	E:

Package pack1.pack2;
Public class Ram
{
public void m1()
{
 	
System.out.println("Hell o");
}
}
	Package pack3.pack4;
	Import pack1.pack2.
	Ram
	pblic class Sham
	{
	Public void m1()
	{
	Ram r=new Ram();
	r.m1();
	System.out.println("Hiiii i");
	}
	}
		Package pack3.pack.Sham;
		Public class Desha
		{
		Public static void main(-)
		{
		Sham s=new Sham();
		s.m1();
		System.out.println("Weldon e");
		}
		}

C:\> java –d.Ram //valid

D:\> java-d.Sham.java //C.E.=cannot find symbol

Symbol:class Ram

Location:class Sham

D:\>java-cp c: -d.Sham.java //valid

E:\>javac Desha.java // C.E.= cannot find symbol

Symbol:class Sham

Location:class Desha

E:\>javac-cp D:Desha.java

E:\>java Desha //R.E.=NoClassDefoundError:Sham

E:\>java-cp D:Desha //R.E.=NoClassDefoundError:Desha

E:\>java-cp ;D:Desha //R.E.=NoClassDefoundError:Ram

E:\>java-cp E: ;D:;C:;Desha

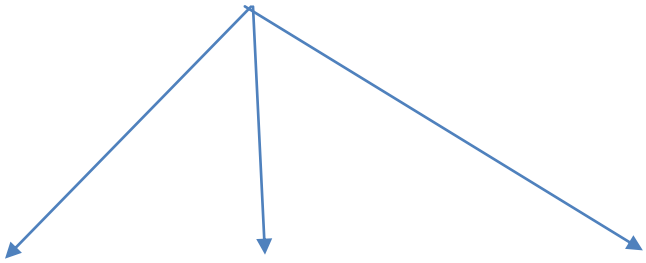
Notes :-

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 location is very impotance for the required .class file ,JVM will always search the location.

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



C: D: E:

class Priya
{
Public static void main(-)
{
System.out.println("C:Priya");
}
}
	class Priya
	{
	Public static void main(-)
	{
	System.out.println("D:Priya");
	}
	}
		class Priya
		{
		Public static void main(-)
		{
		System.out.println("E:Priya");
		}
		}

C:\> javac Priya.java //valid

D:\> javac Priya.java //valid

E:\> javac Priya.java //valid

C:\> java Priya //valid

Output: c:Priya

D:\>java-cp C:;D:;E: Priya

o/p= C:Priya

D:\>java-cp E:;D:;C: Priya

o/p= E:Priya

D:\>java-cp D:;E:;C: Priya

o/p= D:Priya

JAR File

- If several dependent files are available then it is never recommanded to set each class file individually in the classpath we have to group all those. Class file into a single I/P file and we have to make that Zip file available in the class path.This zip file is nothing but JAR file.

Example:

To denelope a servlet all required .class file 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 archive file)**
   It contains a group of .class files
2) **war : (web archive file)**
   it represent a web application which may contain servlets,JSPS,HTML,CSS.JavaScript,etc…….
3) **ear : (enterprise archive file)**
   It represent an enterprise application which may conatins Servlets,JSPS,EJBS,JNS Components etc.

➢ **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 of contains of a jar file:**
   Jar-tvf durga.jar

Example:

```
public class Durga
  {
    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 Durga.java
c:\> jar – cvf Durga.jar durga.class

class Desha
{
  public static void main(String args[])
  {
    System.out.println(“durga.add(10,20)”);
    System.out.println(“durga.multipy(10,20)”);
  }
a}
D:\> javac Desha.java     //invalid
D:\>javac –cp c:Desha.java   //invalid
D:\>  javac –cp c:\Desha.java    //valid
D:\> javac –cp;c:\ Desha.java
```

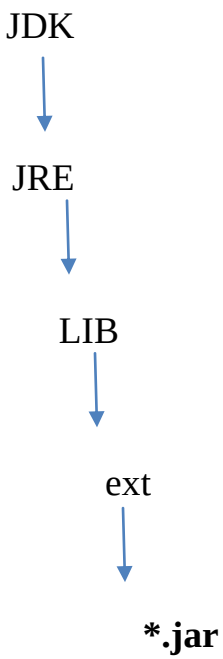
O/P: 2000 &400

Note:

Whenever we are placing a jar file in the classpath compulsory name of the jar file we should include,just Location is not enough.

- **ShortCut way to place jar file:-**

- If we are placing the jar file in the following location then it is not required to set classpath explicitly bydefault it is available to JVM & Java compiler.



System Properties

- For every system persistence information will be maintain in the form of system properties .These may include a name, virtual machine version, user country..etc
- We can get System properties by using getProperties () method of System class

Example:

Demo program to print an 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 option

Que) JDK vs JRE vs JVM

- **JDK(Java Development 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.

- **JDK(Java Virtual Machine):-**
  The 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 developess machine to install JDK.

Difference Between Path And Classpath

- **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 wan't be run.

- **Path**

  - we can use path variable to describe the location where reguired binary executable are available .
  - if we are not setting path variable then java & javac commands wan't work.
  - If m1() return type is void,then we will get compiletimeError saying"void type not allowed here"

4) Various Runtime Flags

1) –ea: To enable assertions in every non-System class.
2) –enableassertions:- It is Exactly same as -ea
3) –da :- To disable assertion in every non-system class.
4) –disableassertions:- same as –da.
5) –esa:- To enabled assertions in every system class.
6) –enabledSystemassertions:- it is exactly same as –esa.
7) –dsa:- To disable assertions in nevery System class.
8) –disableSystemassertions:- it is same as –dsa.

Example:-1)

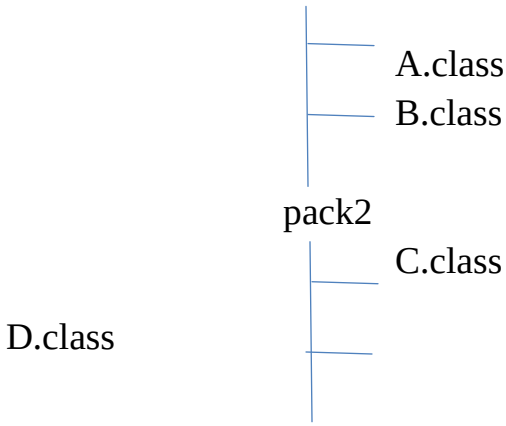
java –ea –esa –da –esa –ea –dsa

we can use these flag in together & all these flags executed from Left To Right

Example:-2)

1)java –ea:pack1.A
2)java –ea:pack1.B -ea:pack1 pack2.D
3) java –ea -da packB

Example:-3)



- TO anable assertion in only A class
  1) Java -ea:pack1.A
- TO anable assertion in both B & D class
  2) Java -ea:pack1.B -ea:pack1:pack2.D
- TO anable assertion in every non-System class except B
  3) Java -ea:pack1. –da:pack1.B
- TO anable assertion in every class of pack1 & its sub except sub packages
  4) Java -ea:pack1.
- TO anable assertion in every where with in pack1 except pack2
  5) Java -ea:pack1. –da:pack1,pack2

6) Appropriate & Inappopriate use of asserting

1) It is always Inappropriate to mix progamimg logic with assert statement because there is no execution of assert statement at runtime.

Example:

Withdraw(int x)
{
If(x<100)
{
Throw new IAE();
}
} //proper way
	Withdraw(int x)
	{
	assert(x<100)
	} //unproperway	