Java means DURGA SOFT..

CORE JAVA

Material

Java™

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

INTERNATIONALIZATION (i18N)

I18N enables the application to support in different languages.

- *Internationalization is also called as i18n because in between I & n 18 words are present.*
- *By using Locale class and ResourceBundle class we are enable I18n on the application.*
- *Local is nothing but language + country.*
- *For making your application to support I18n we need to prepare local specific properties file it means for English one properties file &hindi one properties file …etc.*
- *The property file format is key = value*
- *The properties file name fallowed pattern bundlenamewith language code and country code.*
    - *ApplicationMessages_en_US.properties.*
- *In single web application contains different properties file all the properties files key must be same and values are changed local to Locale.*

Java.util.Locale:-

- *Locale Object is decide properties file based on argument you passed and then it display locale specific details based on Properties file entry.*

    Locale l = new Locale(args[0],args[1]);

    Locale l = new Locale(en,US);

D:\5batch>javap java.util.Locale

Compiled from "Locale.java"
public final class java.util.Locale extends java.lang.Object i
public static final java.util.Locale ENGLISH;
public static final java.util.Locale FRENCH;
public static final java.util.Locale GERMAN;
public static final java.util.Locale ITALIAN;
public static final java.util.Locale JAPANESE;
public static final java.util.Locale KOREAN;
public static final java.util.Locale CHINESE;
public static final java.util.Locale SIMPLIFIED_CHINESE;
public static final java.util.Locale TRADITIONAL_CHINESE;
public static final java.util.Locale FRANCE;
public static final java.util.Locale GERMANY;
public static final java.util.Locale ITALY;
public static final java.util.Locale JAPAN;
public static final java.util.Locale KOREA;
public static final java.util.Locale CHINA;
public static final java.util.Locale PRC;

public static final java.util.Locale TAIWAN;
public static final java.util.Locale UK;
public static final java.util.Locale US;
public static final java.util.Locale CANADA;
public static final java.util.Locale CANADA_FRENCH;

Sample Language Codes

Language Code	Description
de	German
en	English
fr	French
ru	Russian
ja	Japanese
jv	Javanese
ko	Korean
zh	Chinese

To get particular language and country code use fallowing example:-

```
importjava.util.*;
class Test
{       public static void main(String[] args)
        {       Locale l = Locale.FRANCE;
                System.out.println(l.getLanguage());
                System.out.println(l.getCountry());
        }
}
```

D:\5batch>java Test
fr
FR

Java.util.ResourceBundle:-

Creates ResourceBundle object by passing Local object then by using ResourceBoundle we are able to get data form properties file that is decide by Locale.

- *It is possible to create ResourceBundle Object without specifying Locale it will take default properties file with default language.*
  ResourceBundle bundle1 = ResourceBundle.getBundle("Application");
- *It is possible to create ResourceBundle Object by specifying default Locale object.*
  ResourceBundle bundle2 = ResourceBundle.getBundle("Application",Locale.FRANCE);
- *It is possible to create ResourceBundle object by creating new user Locale Object*
  ResourceBundle bundle3 = ResourceBundle.getBundle("Application",new Locale("ratan","RATAN"));

Application 1:-

<u>Steps to design application:-</u>

Step-1:- prepare properties files to support different languages and countries.

Application.properties — *default properties file(base properties file)*
Application_fr_FR.properties — *French properties file*
Allication_ratan_RATAN.properties — *Ratan country properties file*

Step 2:- create locale object it identified particular language and country and it decides execution of properties file.

Locale l = new Locale("en","US");

The above statement specify language is English and country united states

Locale l = new Locale("fr","CA");
Locale x = new Locale("fr","FR");

The above two locales specifies France language in Canada& France

Instead of hard coding language name and country name get the values from command prompt at runtime.

```
Public static void main(String[ ] args)
{       Locale l = new Locale(args[0],args[1]);
}
```

D:\5batch>java Test fr FR

Step 3:-create ResourceBundle by passing Locale object.

//if no local is Matched this property file is executed [default property file]

ResourceBundle bundle1 = ResourceBundle.getBundle("Application");

//it create ResourceBundle with local that is already defined [France properties file]

ResourceBundle bundle2 = ResourceBundle.getBundle("Application",Locale.FRANCE);

Step 4:- fetch the text form ResourceBundle

String msg = Bundle.getString("wish");
System.out.println(msg);

Application.properties:-
countryname = USA
lang = eng

Application_fr_FR.properties:-
countryname = canada
lang = france

Allication_ratan_RATAN.properties:-
countryname=Ratan
lang= ratan

Test.java:-

```
importjava.util.*;
class Test
{       public static void main(String[] args)
        {
//if no local is Matched this property file is executed
ResourceBundle bundle1 = ResourceBundle.getBundle("Application");
//it create ResourceBundle with local that is already defined
Locale l1 = Locale.FRANCE;
ResourceBundle bundle2 = ResourceBundle.getBundle("Application",l1);
//it creates ResourceBundle with new user created Locale
Locale l2 = new    Locale("ratan","RATAN")
ResourceBundle bundle3 = ResourceBundle.getBundle("Application",l2);
System.out.println(bundle1.getString("countryname")+"--"+bundle1.getString("lang"));
System.out.println(bundle2.getString("countryname")+"--"+bundle2.getString("lang"));
System.out.println(bundle3.getString("countryname")+"--"+bundle3.getString("lang"));
        }
}
```

Output:-

```
D:\5batch>java Test
USA--eng
Canada--france
Ratan--Ratan
```

APPLICATION 2:-

```
importjava.util.*;
class Test
{       public static void main(String[] args)
        {       //creates local object with the help of arguments
                Locale l = new Locale(args[0],args[1]);
        //it creates resource bundle with local passed from as command line arguments
                ResourceBundle bundle = ResourceBundle.getBundle("Application",l);
                System.out.println(bundle.getString("countryname"));
                System.out.println(bundle.getString("lang"));
        }
}
D:\5batch>java Test x y
USA
eng
D:\5batch>java Test fr FR
canada
france
D:\5batch>java Test ratan RATAN
Ratan
ratan
```

Application before internationalization:-

```
importjava.util.*;
class Test
{
        public static void main(String[] args)
        {       System.out.println("hello");
                System.out.println("i like you");
                System.out.println("i hate you");
        }
}
```

We are decide to print this messages in different languages like Germany, French…..etc then we must translate the code in different languages by moving the message out of source code to text file it looks the program need to be internationalized(supporting different languages).

Application.properties:-

```
wish = hello
lovely = i love you
```

angry = i hate you

Application_fr_FR.properties:-
wish = hlloe
lovely = i evol you
angry = i etah you

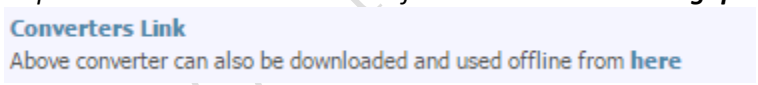
Application_hi_IN.properties:-
wish=\u0c39\u0c46\u0c32\u0c4d\u0c32\u0c4a
lovely=\u0c07 \u0c32\u0c4a\u0c35\u0c46 \u0c2f\u0c4a\u0c09
angry=\u0c07 \u0c39\u0c24\u0c46 \u0c09

```
importjava.util.*;
class Test
{       public static void main(String[] args)
        {       Locale l = new Locale(args[0],args[1]);
                ResourceBundlerb = ResourceBundle.getBundle("Application",l);
                System.out.println(rb.getString("wish"));
                System.out.println(rb.getString("lovely"));
                System.out.println(rb.getString("angry"));
        }
}
```

```
D:\5batch>java Test x y
hello
i love you
i hate you
```

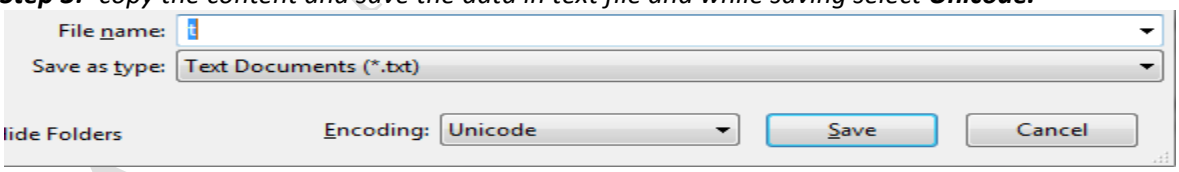
```
D:\5batch>java Test fr FR
hlloe
ievol you
ietah you
D:\5batch>java Test hi IN
??????
? ???? ???
? ??? ?
```

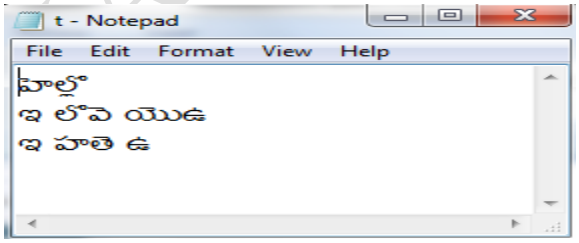
Conversion of any language to Unicode values:-
Step 1:- download Unicode editor from internet ***www.higopi.com***



Step 2:- unzip the file and click on index.html page select language and type the words.

Step 3:- *copy the content and save the data in text file and while saving select* ***Unicode.***

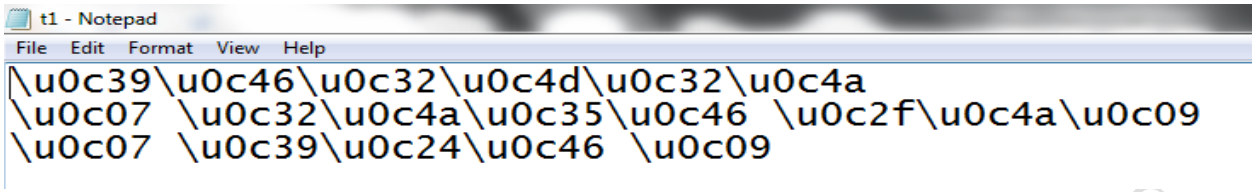




Step 4:- convert the above language to Unicode character format.

Syntax:- native2ascii -encoding encoding-name source-file destination-file

D:\>native2ascii -encoding unicode t.txt output.txt



Application :-

Application.properties:-

wish = hello
lovely = i love you
angry = i hate you

Application_fr_FR.properties:-

wish = hlloe
lovely = i evol you
angry = i etah you

Application_tl_IN.properties:-

wish=\u0c39\u0c46\u0c32\u0c4d\u0c32\u0c4a
lovely=\u0c07 \u0c32\u0c4a\u0c35\u0c46 \u0c2f\u0c4a\u0c09
angry=\u0c07 \u0c39\u0c24\u0c46 \u0c09

Test.java:-

```
importjava.util.*;
importjava.awt.*;
class Test
{       public static void main(String[] args)
        {       Locale l = new Locale(args[0],args[1]);
                ResourceBundle b = ResourceBundle.getBundle("Application",l);
                Frame f = new Frame();          //to create frame
                f.setVisible(true);         //to provide visibility to frame
                f.setSize(300,75);//to align the frame set bounds
                f.setLayout(new FlowLayout());//to set the frame proper format
                //creation of buttons with labels
                Button b1 = new Button(b.getString("wish"));
                Button b2 = new Button(b.getString("lovely"));
                Button b3 = new Button(b.getString("angry"));
                //adding buttons into frame
                f.add(b1);
                f.add(b2);
                f.add(b3);
        }
}
```

Test.java:- example

```
importjava.util.*;
public class Test {
static public void main(String[] args) {
    String language;
    String country;
```

```
    Locale currentLocale;
ResourceBundle messages;
if (args.length != 2)
{       language = new String("en");
country = new String("US");
    }
        else
        {   language = new String(args[0]);
country = new String(args[1]);
    }
currentLocale = new Locale(language, country);
messages = ResourceBundle.getBundle("Application", currentLocale);
System.out.println(messages.getString("wish"));
System.out.println(messages.getString("lovely"));
System.out.println(messages.getString("angry"));
  }
}
D:\5batch>java Test
hello
i love you
i hate you
D:\5batch>java Test x y
hello
i love you
i hate you
D:\5batch>java Test tl IN
??????
? ???? ???
? ??? ?
D:\5batch>java Test fr FR
hlloe
ievol you
ietah you
```

Example :- display Date in different Locale.

DateFormat.DEFAULT,
DateFormat.SHORT,
DateFormat.MEDIUM,
DateFormat.LONG,
DateFormat.FULL

Sample Date Formats

Style	U.S. Locale	French Locale
DEFAULT	Jun 30, 2009	30 juin 2009
SHORT	6/30/09	30/06/09
MEDIUM	Jun 30, 2009	30 juin 2009
LONG	June 30, 2009	30 juin 2009
FULL	Tuesday, June 30, 2009	mardi 30 juin 2009

Test.java:-

```
importjava.util.*;
importjava.text.DateFormat;
class Test
{       public static void main(String[] args)
        {
Date d = new Date();
//default locale en US
DateFormat df1 = DateFormat.getDateInstance(DateFormat.DEFAULT,Locale.getDefault());
System.out.println(df1.format(d));
//date of fresh
DateFormat df2 =DateFormat.getDateInstance(DateFormat.MEDIUM,Locale.FRENCH);
System.out.println(df2.format(d));
//date of Italy
DateFormat df3 =DateFormat.getDateInstance(DateFormat.SHORT,Locale.ITALY);
System.out.println(df3.format(d));
        }
};
```

```
D:\5batch>java Test
Nov 21, 2014
21 nov. 2014
21/11/14
```

Example on time format:-

Sample Time Formats

Style	U.S. Locale	German Locale
DEFAULT	7:03:47 AM	7:03:47
SHORT	7:03 AM	07:03
MEDIUM	7:03:47 AM	07:03:07
LONG	7:03:47 AM PDT	07:03:45 PDT
FULL	7:03:47 AM PDT	7.03 Uhr PDT

```
importjava.util.*;
importjava.text.*;
class Test
```

```
{       public static void main(String[] args)
        {Date d = new Date();
DateFormat df1 = DateFormat.getTimeInstance(DateFormat.DEFAULT,Locale.getDefault());
System.out.println(df1.format(d));
DateFormat df2 = DateFormat.getTimeInstance(DateFormat.MEDIUM,Locale.FRENCH);
System.out.println(df2.format(d));
DateFormat df3 = DateFormat.getTimeInstance(DateFormat.SHORT,Locale.ITALY);
System.out.println(df3.format(d));
        }
};
```

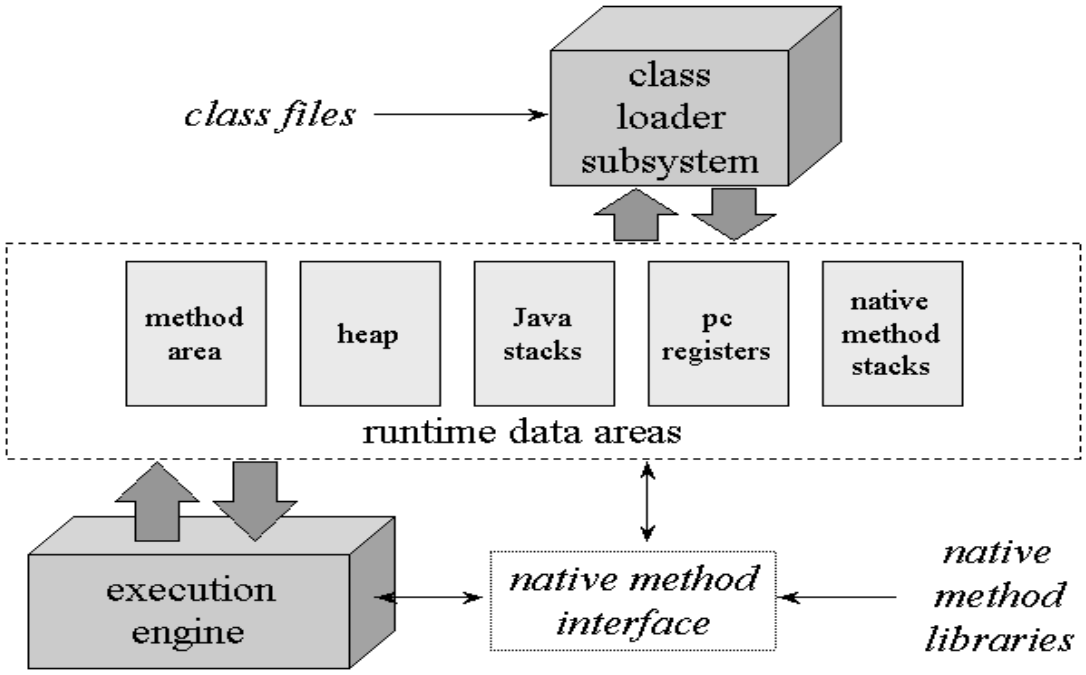
Example on both data and Time format:-

Sample Date and Time Formats

Style	U.S. Locale	French Locale
DEFAULT	Jun 30, 2009 7:03:47 AM	30 juin 2009 07:03:47
SHORT	6/30/09 7:03 AM	30/06/09 07:03
MEDIUM	Jun 30, 2009 7:03:47 AM	30 juin 2009 07:03:47
LONG	June 30, 2009 7:03:47 AM PDT	30 juin 2009 07:03:47 PDT
FULL	Tuesday, June 30, 2009 7:03:47 AM PDT	mardi 30 juin 2009 07 h 03 PDT

```
importjava.util.*;
importjava.text.*;
class Test
{       public static void main(String[] args)
        {       Date d = new Date();
DateFormat df1 =  DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL,Locale.getDefault());
System.out.println(df1.format(d));
DateFormat df2 =  DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL,Locale.FRENCH);
System.out.println(df2.format(d));
        }
};
```

JVM Architecture:-



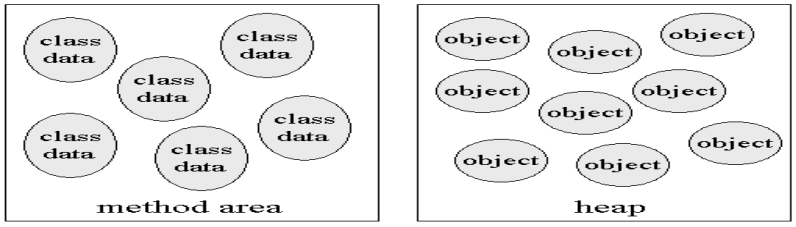
Class loader subsystem:-

1. *It is used to load the classes and interfaces.*
2. *It verifies the byte code instructions.*
3. *It allots the memory required for the program.*

Runtime data area:-this is the memory resource used by the JVM and it is 5 types

Method Area:-*It is used to store the class data and method data.*

Heap area:-*It is used to store the Objects.*

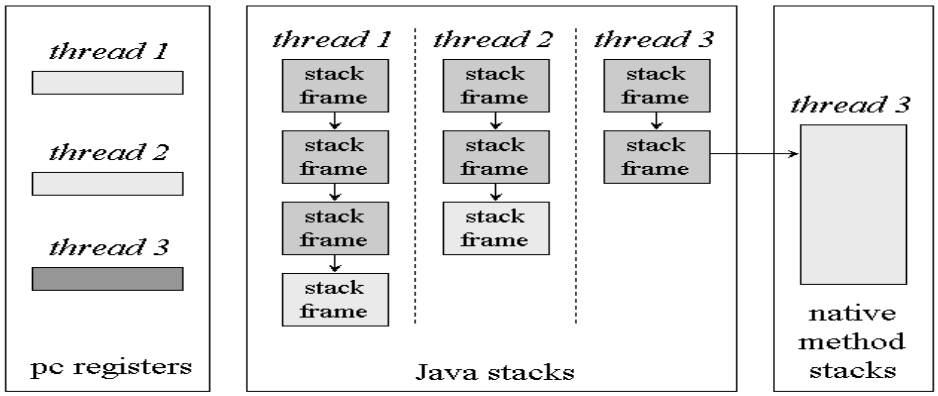


Runtime data areas shared among all threads.

Java stacks:-

- *Whenever new thread is created for each and every new thread the JVM will creates PC(program coubter) register and stack.*

- *If a thread executing java method the value of pc register indicates the next instruction to execute.*
- *Stack will stores method invocations of every thread. The java method invocation includes local variables and return values and intermediate calculations.*
- *The each and every method entery will be stored in stack. And the stack contains grounp of enteries and each and every entry stored in one stack frame hence stack is group of stack frames.*
- *Whenever the method completes the entry is automatically deleted from the stach so whatever the functionalities declared in method it is applicable only for respective methods.*
  Java native method stack is used to store the native methods invocations.



Runtime data areas exclusive to each thread.

Native method interface:-

Native method interface is a program that connects native methods libraries (C header files) with JVM for executing native methods.

Native method library: It contains native libraries information.

Execution engine:-

It is used to execute the instructions are available in the methods of loaded classes. It contains JIT(just in time compiler) and interpreter used to convert byte code instructions into machine understandable code.

Modifiers summary:-

- ***In java no concept like "access specifiers and access modifiers"andonly one concept is there modifiers concept.***
- ***How many Modifiersinjava means don't say 3 or 4 or 5 ,in java 11 modifiers are there.***
- ***The default modifier in java is "default".***
- ***The most restricted modifier in java is private (only with in the class).***
- ***The most accessable modifier in java is public (all package can access)***
- ***The only one modifier applicable to local variables is "final".***

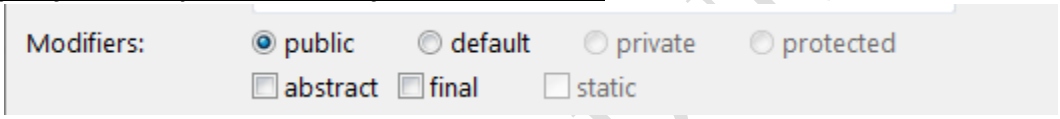
Proof 1:-

```
private class Test
{       public static void main(String[] args)
        {
        }
}
```

Compilation Error:-

```
D:\morn11>javac Test.java
Test.java:1: modifier private not allowed here
private class Test
```

proof 2:- in eclips IDE shows information like this.



modifier	***classes***	***methods***	***variables***
public	*yes*	*yes*	*yes*
private	*no*	*yes*	*yes*
default	*yes*	*yes*	*yes*
protected	*no*	*yes*	*yes*
final	*yes*	*yes*	*yes*
abstract	*yes*	*yes*	*no*
strictfp	*yes*	*yes*	*no*
transient	*no*	*no*	*yes*
native	*no*	*yes*	*no*
static	*no*	*yes*	*yes*
synchronized	*no*	*yes*	*no*
volatile	*no*	*no*	*yes*

Java is not a pure object oriented programming language:-

1)java supporting primitive datatypes there are not a objects. To represent these primitives in the forn of objects java having concept like Wrapper classes.

```
Int a=10;
Boolean b=true;
```

2)without creation of object we are able to access static members.

```
class Test
{       static void m1()
        {   System.out.println("hi ratan");
        }
        public static void main(String[] args)
        {       Test.m1();
        }
}
```

3) java is not supporting oops concepts like multiple inheritance& hybrid inheritance.

Class A extends B,C===➔error

Different approaches to create objects in java:-

1. *By using new operator.*
2. *by using clone() method.*
3. *without using new operator by using **String str="ratan"; [by using String content].***
4. *at the time of deserialization we are getting the data from file we are stored in object form.*
5. *By using factory method*
    a. *Instance factory method*
    b. *Static factory method*
6. *By using newInstance method. (used by servers)……etc*