InternationalizatioN(I18N)

The process of designing web applications in such a way that which provides supports for various countries, various languages and various currencies automatically without performing any change in the application, is called InternationalizatioN (I18N).

If request is coming from India, then response should be India people understandable form. Similarly, if request is coming from USA, then response should be USA people understandable form.

We can implement InternationalizatioN by using following three classes:

1. Locale
2. NumberFormat
3. DateFormat

Locale

- A Locale Object represents Geographic Location (Country) or Language or Both.
- We can create Locale Object to represent India. And we can also create a Locale Object to represent English Language.
- Locale class presents in java.util package.
- Locale class is a final class and it is a direct child class of Object Class. It implements Serializable and Cloneable Interfaces.

Constructors

1. Locale l = new Locale(String language);
2. Locale l = new Locale(String language, String country);

Locale class already defined some constants to define some standard Locales. We can use these Constants Directly.
Example:
Locale.US, Locale.UK, Locale.ENGLISH, Locale.ITALY etc.

Important Methods of Locale Class

public static Locale getDeafault()	Returns default Locale
public static void setDefault(Locale l)	Set Default Locale
public String getCountry()	Returns Country Short Name e.g. US
public String getLanguage ()	Returns Language Short Name e.g. ENG
public String getDispalyCountry()	Returns Country Full Name e.g. United States
public String getDisplayLanguage()	Returns Language Full Name e.g. English
public static String[] getISOLanguages()	Returns an array of All ISO Supported Languages.
public static String[] getISOCountries()	Returns an array of All ISO Supported Countries.
public static Locale[] getAvailableLocales()	Returns an array of All available Locales.

Sample Program Snippet

Main.java

Run

```
1  import java.util.*;
2  class LocaleDemo {
3      public static void main(String[] args) {
4          Locale l1 = Locale.getDefault();
5          System.out.println("Default Language : " + l1.getLanguage());
6          System.out.println("Default Display Language : " + l1.getDisplayLanguage());
7          System.out.println("Default Country : " + l1.getCountry());
8          System.out.println("Default Display Country : " + l1.getDisplayCountry());
9
10         Locale l2 = new Locale("pa","IN");
11         Locale.setDefault(l2);
12         System.out.println("New Default Language : " + Locale.getDefault().getLanguage());
13         System.out.println("New Default Display Language : " + Locale.getDefault().getDisplayLanguage
               ());
14         System.out.println("New Default Country : " + Locale.getDefault().getCountry());
15         System.out.println("New Default Display Country : " + Locale.getDefault().getDisplayCountry());
16         System.out.println("All ISO Supported Languages : ");
17         System.out.println(Arrays.asList(Locale.getISOLanguages()));
18         System.out.println("All ISO Supported Countries : ");
19         System.out.println(Arrays.asList(Locale.getISOCountries()));
20         System.out.println("All Available Locales : ");
21         System.out.println(Arrays.asList(Locale.getAvailableLocales()));
22         System.out.println("Total ISO Supported Languages : " + Arrays.asList(Locale.getISOLanguages
               ()).size());
23         System.out.println("Total ISO Supported Countries : " + Arrays.asList(Locale.getISOCountries
               ()).size());
24         System.out.println("Total Available Locales : " + Arrays.asList(Locale.getAvailableLocales
               ()).size());
25     }
26 }
```

Output

Clear

```
java -cp /tmp/ir4Hw2hHni LocaleDemo
Default Language : en
Default Display Language : English
Default Country : US
Default Display Country : United States
New Default Language : pa
New Default Display Language : ??????
New Default Country : IN
New Default Display Country : ????
All ISO Supported Languages :
[aa, ab, ae, af, ak, am, an, ar, as, av, ay, az, ba, be, bg, bh, bi, bm, bn, bo, br, bs, ca, ce, ch, co, cr
    , cs, cu, cv, cy, da, de, dv, dz, ee, el, en, eo, es, et, eu, fa, ff, fi, fj, fo, fr, fy, ga, gd, gl,
    gn, gu, gv, ha, he, hi, ho, hr, ht, hu, hy, hz, ia, id, ie, ig, ii, ik, in, io, is, it, iu, iw, ja, ji,
    jv, ka, kg, ki, kj, kk, kl, km, kn, ko, kr, ks, ku, kv, kw, ky, la, lb, lg, li, ln, lo, lt, lu, lv, mg,
    mh, mi, mk, ml, mn, mo, mr, ms, mt, my, na, nb, nd, ne, ng, nl, nn, no, nr, nv, ny, oc, oj, om, or, os,
    pa, pi, pl, ps, pt, qu, rm, rn, ro, ru, rw, sa, sc, sd, se, sg, si, sk, sl, sm, sn, so, sq, sr, ss, st,
    su, sv, sw, ta, te, tg, th, ti, tk, tl, tn, to, tr, ts, tt, tw, ty, ug, uk, ur, uz, ve, vi, vo, wa, wo,
    xh, yi, yo, za, zh, zu]
All ISO Supported Countries :
[AD, AE, AF, AG, AI, AL, AM, AO, AQ, AR, AS, AT, AU, AW, AX, AZ, BA, BB, BD, BE, BF, BG, BH, BI, BJ, BL, BM
    , BN, BO, BQ, BR, BS, BT, BV, BW, BY, BZ, CA, CC, CD, CF, CG, CH, CI, CK, CL, CM, CN, CO, CR, CU, CV,
    CW, CX, CY, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, EH, ER, ES, ET, FI, FJ, FK, FM, FO, FR, GA, GB, GD,
    GE, GF, GG, GH, GI, GL, GM, GN, GP, GQ, GR, GS, GT, GU, GW, GY, HK, HM, HN, HR, HT, HU, ID, IE, IL, IM,
    IN, IO, IQ, IR, IS, IT, JE, JM, JO, JP, KE, KG, KH, KI, KM, KN, KP, KR, KW, KY, KZ, LA, LB, LC, LI, LK,
    LR, LS, LT, LU, LV, LY, MA, MC, MD, ME, MF, MG, MH, MK, ML, MM, MN, MO, MP, MQ, MR, MS, MT, MU, MV, MW,
    MX, MY, MZ, NA, NC, NE, NF, NG, NI, NL, NO, NP, NR, NU, NZ, OM, PA, PE, PF, PG, PH, PK, PL, PM, PN, PR,
    PS, PT, PW, PY, QA, RE, RO, RS, RU, RW, SA, SB, SC, SD, SE, SG, SH, SI, SJ, SK, SL, SM, SN, SO, SR, SS,
    ST, SV, SX, SY, SZ, TC, TD, TF, TG, TH, TJ, TK, TL, TM, TN, TO, TR, TT, TV, TW, TZ, UA, UG, UM, US, UY,
    UZ, VA, VC, VE, VG, VI, VN, VU, WF, WS, YE, YT, ZA, ZM, ZW]
```

```
All Available Locales :
[, nn, ar_JO, bg, kea, nds, zu, am_ET, fr_DZ, ti_ET, bo_CN, hsb, qu_EC, ta_SG, lv, en_NU, zh_SG_#Hans,
    en_MS, en_GG, en_JM, vo, kkj, sr_ME, sv_SE, es_BO, dz_BT, mer, sah, en_ZM, fr_ML, br, ha_NG, ar_SA,
    fa_AF, dsb_DE, sk, os_GE, ml, en_MT, en_LR, ar_TD, en_GH, en_IL, sv, cs, el, tzm_MA, af, sw_UG, ses_ML,
    smn, tk_TM, sr_ME_#Cyrl, ar_EG, dsb, lkt_US, vai_LR_#Latn, ji_001, yo_NG, se_NO, khq, sw_CD, vo_001,
    en_PW, pl_PL, fil_PH, it_VA, sr_CS, ne_IN, es_PH, es_ES, es_CO, bg_BG, ji, ar_EH, bs_BA_#Latn, en_VC,
    nds_DE, nb_SJ, es_US, agq, hsb_DE, en_US_POSIX, en_150, ar_SD, en_KN, ha_NE, pt_MO, ebu, ro_RO,
    zh__#Hans, lb_LU, sr_ME_#Latn, es_GT, so_KE, dje_NE, bas_CM, fr_PM, ar_KM, fr_MG, no_NO_NY, es_CL, mn,
    agq_CM, kam_KE, teo, tr_TR, eu, fa_IR, en_MO, wo, shi__#Tfng, en_BZ, sq_AL, ar_MR, es_DO, ru, twq_NE,
    az, nmg_CM, fa, kl_GL, en_NR, nd, kk, az__#Cyrl, en_MP, en_GD, tk, hy, shi__#Latn, en_BW, en_AU, en_CY,
    kab_DZ, kde_TZ, ta_MY, ti_ER, nus_SS, en_RW, nd_ZW, sv_FI, ksb, luo, lb, ne, en_IE, zh_SG, ln_CD, en_KI
    , nnh_CM, om_ET, no, ja_JP, my, ka, ar_IL, mgh, or_IN, fr_MF, shi, kl, en_SZ, rwk_TZ, zh, es_PE, mgh_MZ
    , saq, az__#Latn, ta, en_GB, lag, zh_HK_#Hant, ar_SY, ksf_CM, bo, kk_KZ, es_PA, tt_RU, om_KE, ar_PS,
    en_AS, fr_VU, zh_TW, bez, kln, fr_MC, kw, pt_MZ, fr_NE, vai__#Latn, ksb_TZ, ksh, ur_IN, ln, en_JE,
    gsw_CH, ln_CF, en_CX, luy_KE, pt, en_AT, gl, kkj_CM, sr__#Cyrl, yue_CN_#Hans, es_GQ, kn_IN, ar_YE, to,
    en_SX, ga, qu, ru_KZ, en_TZ, et, en_PR, mua, ko_KP, in, ps, sn, nl_SR, rof, en_BS, km, zgh, fr_NC, be,
    gv, es, dua, gd_GB, jgo, nl_BQ, fr_CM, gsw, uz_UZ_#Cyrl, pa_IN_#Guru, en_KE, guz, mfe, asa_TZ, teo_UG,
    ja, fr_SN, or, brx, fr_MA, pt_LU, fr_BL, en_NL, mgo_CM, ln_CG, te, sl, ko_KR, el_CY, mr_IN, ha, es_MX,
    lrc_IR, gsw_FR, es_HN, hu_HU, ff_SN, sbp, sq_MK, sr_BA_#Cyrl, fi, uz, bs__#Cyrl, et_EE, sr__#Latn,
    en_SS, sw, bo_IN, fy_NL, ar_OM, tr_CY, nmg, rm, en_MG, fr_BI, uz_UZ_#Latn, bn, dua_CM, de_IT, lrc_IQ,
    vai__#Vaii, kn, fr_TN, sr_RS, de_CH, bn_BD, nnh, fr_PF, en_ZA, gu, pt_GQ, vun_TZ, jmc_TZ, en_TV, lo,
    fr_FR, en_PN, en_MH, fr_BJ, zh__#Hant, cu_RU, zh_HK_#Hans, nl_NL, sah_RU, en_GY, ps_AF, bs__#Latn, ky,
    mas, dyo_SN, os, bs_BA_#Cyrl, nl_CW, ar_DZ, sk_SK, pt_CH, fr_GQ, ff_CM, am, en_NG, fr_CI, ki_KE, en_PK,
    zh_CN, en_LC, rw, brx_IN, wo_SN, iw, gv_IM, mk_MK, en_TT, dav, sl_SI, fr_HT, te_IN, nl_SX, lrc, ses, ce
    , fr_CG, fr_BE, jgo_CM, mt_MT, es_VE, mg, mr, mer_KE, ko, nds_NL, en_BM, nb_NO, ak, seh, kde, dz,
    kea_CV, mgo, vi_VN, en_VU, en_US, to_TO, mfe_MU, seh_MZ, fr_BF, pa__#Guru, it_SM, fr_YT, gu_IN, ii_CN,
    pa_PK_#Arab, ast, fr_RE, fi_FI, yue__#Hans, ca_FR, sr_BA_#Latn, bn_IN, fr_GP, pa, zgh_MA, uk_UA, fr_DJ,
    rn, tg, rwk, hu, fr_CH, en_NF, twq, ha_GH, sr_XK_#Cyrl, bm, ar_SS, en_GU, nl_AW, de_BE, en_AI, en_CM,
    xog_UG, cs_CZ, tr, ca_ES, cgg, rm_CH, nyn_UG, ru_MD, ms_MY, ta_LK, ksf, en_TO, cy, en_PG, fr_CF, pt_TL,
    sq, fr, tg_TJ, en_ER, qu_PE, sr_BA, es_PY, de, es_EC, kok_IN, lg_UG, zu_ZA, fr_TG, sr_XK_#Latn, en_PH,
    ig_NG, fr_GN, prg_001, cgg_UG, zh_MO_#Hans, ksh_DE, lg, ru_RU, se_FI, ff, en_DM, en_CK, sd, ar_MA,
    ga_IE, en_BI, en_AG, fr_TD, en_WS, fr_LU, ebu_KE, bem_ZM, xog, ewo_CM, fr_CD, so, rn_BI, en_NA, ar_ER,
    kab, ms, nus, sn_ZW, prg, iw_IL, ug, es_EA, th_TH_TH_#u-nu-thai, hi, fr_SC, ca_IT, lag_TZ, en_SL,
    teo_KE, no_NO, ca_AD, zh_MO_#Hant, en_SH, vai, qu_BO, haw_US, vi, fr_CA, de_LU, sq_XK, dyo, en_KY, mt,
    it_CH, de_DE, si_LK, luo_KE, en_DK, yav, so_DJ, lt_LT, it_IT, eo, kam, ar_SO, en_ZW, ro, eo_001, ee,
    en_UM, nn_NO, fr_MU, pl, se_SE, en_TK, en_SI, mua_CM, ur, uz__#Arab, vai_LR_#Vaii, saq_KE, se, pt_GW,
    lo_LA, chr, ar_LB, af_ZA, ms_SG, ee_TG, ln_AO, be_BY, ff_GN, yue__#Hant, in_ID, es_BZ, ar_AE, hr_HR,
    luy, as, rof_TZ, it, pt_CV, ks_IN, uk, my_MM, ur_PK, mn_MN, da_DK, en_FM, es_PR, wae_CH, mzn, en_BE, ii
    , tt, fr_WF, ru_BY, mzn_IR, naq, fo_DK, en_SG, ee_GH, ar_BH, kln_KE, tzm, fur, om, hi_IN, en_CH, asa,
    yo_BJ, fo_FO, ast_ES, fr_KM, bez_TZ, fr_MQ, en_SD, es_AR, en_MY, ja_JP_JP_#u-ca-japanese, es_SV, pt_BR,
    ml_IN, sbp_TZ, fil, en_FK, uz__#Cyrl, is_IS, yue_HK_#Hant, hy_AM, en_GM, en_DG, fo, ne_NP, hr, pt_ST,
    ak_GH, lt, uz_AF_#Arab, fur_IT, ta_IN, ccp, en_SE, fr_GF, lkt, zh_CN_#Hans, is, es_419, si, pt_AO,
    en_001, en, guz_KE, gsw_LI, ccp_BD, es_IC, ca, ru_KG, fr_MR, ar_TN, ks, zh_TW_#Hant, bm_ML, kw_GB,
    ug_CN, as_IN, es_BR, zh_HK, khq_ML, sw_KE, en_SB, th_TH, rw_RW, chr_US, shi_MA_#Tfng, ar_IQ, nyn, yue,
    jmc, en_MW, naq_NA, mk, en_IO, ar_QA, en_DE, pa__#Arab, en_CC, bs, ro_MD, en_FI, pt_PT, fy, az_AZ_#Cyrl
    , th, dav_KE, ckb_IQ, shi_MA_#Latn, es_CU, ar, en_SC, en_VI, haw, eu_ES, en_UG, en_NZ, dje, es_UY, bas,
    mas_KE, ru_UA, sg_CF, el_GR, yav_CM, uz__#Latn, sg, da_GL, en_FJ, de_LI, en_BB, km_KH, smn_FI, hr_BA,
    de_AT, ckb_IR, nl, lu_CD, ca_ES_VALENCIA, ar_001, so_SO, lv_LV, ckb, es_CR, fr_GA, ar_KW, sr, ar_LY,
    sr_RS_#Cyrl, bem, en_MU, da, wae, gl_ES, en_IM, az_AZ_#Latn, en_LS, ig, en_HK, en_GI, ce_RU, en_CA, gd,
    ka_GE, fr_SY, sw_TZ, fr_RW, so_ET, nl_BE, ar_DJ, mg_MG, cy_GB, en_VG, cu, os_RU, sr_RS_#Latn, en_TC,
    ky_KG, sv_AX, af_NA, vun, en_IN, lu, ki, yo, es_NI, nb, ff_MR, sd_PK, mas_TZ, ti, kok, ewo, ms_BN,
    ccp_IN, br_FR]
Total ISO Supported Languages : 188
Total ISO Supported Countries : 249
Total Available Locales : 748
```

NumberFormat

- Various Locations follow various styles to represent a Java Number.
  Example:
  Double d = 123456.789 can be represented as:
  IN – 1,23,456.789
  US – 123,456.789
  ITALY – 123.456,789
- We can use NumberFormat class to format or represent the same number differently as per the Locales.
- NumberFormat class is present in java.text package and it is an abstract class.

- We cannot create an Object of NumberFormat Class directly. i.e., NumberFormat nf = new NumberFormat(); is not possible in Java.

Getting NumberFormat Object for Default Locale

NumberFormat Class defines the following methods to get Object for Default Locale:

1. public static NumberFormat getInstance();
2. public static NumberFormat getNumberInstance();
3. public static NumberFormat getCurrencyInstance();
4. public static NumberFormat getPercentInstance();

Here 1 and 2 are exactly Same.

Getting NumberFormat Object for Specific Locale

NumberFormat Class defines the following methods to get Object for Specific Locales which are same as above four methods, only difference is we have to pass the Object of Locale as an argument for that specific Locales:

1. public static NumberFormat getInstance(Locale L);
2. public static NumberFormat getNumberInstance(Locale L);
3. public static NumberFormat getCurrencyInstance(Locale L);
4. public static NumberFormat getPercentInstance(Locale L);

Once we get NumberFormat Object we can call format() or parse() methods on that Object.

1. public String format(long l);
2. public String format(double d);

These two methods convert Java Number to Locale specific String.

3. public Number parse(String s) throws ParseException;

This method converts Locale specific String form to Java Number form.

Sample Java Program to demonstrate the NumberFormat Class Use

```
1  import java.util.*;
2  import java.text.*;
3
4  class NumberFormatDemo {
5      public static void main(String[] args) {
6          double d = 123456.789;
7          NumberFormat n2 = NumberFormat.getNumberInstance(Locale.US);
8          NumberFormat n3 = NumberFormat.getNumberInstance(Locale.ITALY);
9          System.out.println("US Representation of Number : " + n2.format(d));
10         System.out.println("Italy Representation of Number : " + n3.format(d));
11     }
12 }
```

```
java -cp /tmp/ir4Hw2hHni NumberFormatDemo
US Representation of Number : 123,456.789
Italy Representation of Number : 123.456,789
```

```java
import java.util.*;
import java.text.*;

class NumberFormatDemo {
    public static void main(String[] args) {
        double d = 123456.789;
        Locale l = new Locale("pa","IN");
        NumberFormat n1 = NumberFormat.getInstance(l);
        NumberFormat n2 = NumberFormat.getNumberInstance(Locale.US);
        NumberFormat n3 = NumberFormat.getNumberInstance(Locale.UK);

        NumberFormat n4 = NumberFormat.getCurrencyInstance(l);
        NumberFormat n5 = NumberFormat.getCurrencyInstance(Locale.US);
        NumberFormat n6 = NumberFormat.getCurrencyInstance(Locale.UK);

        NumberFormat n7 = NumberFormat.getPercentInstance(l);
        NumberFormat n8 = NumberFormat.getPercentInstance(Locale.US);
        NumberFormat n9 = NumberFormat.getPercentInstance(Locale.UK);

        System.out.println("IN Representation of Number : " + n1.format(d));
        System.out.println("US Representation of Number : " + n2.format(d));
        System.out.println("UK Representation of Number : " + n3.format(d));

        System.out.println("IN Representation of Currency : " + n4.format(d));
        System.out.println("US Representation of Currency : " + n5.format(d));
        System.out.println("UK Representation of Currency : " + n6.format(d));

        System.out.println("IN Representation of Percent : " + n7.format(d));
        System.out.println("US Representation of Percent : " + n8.format(d));
        System.out.println("UK Representation of Percent : " + n9.format(d));
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-ja
IN Representation of Number : 123,456.789
US Representation of Number : 123,456.789
UK Representation of Number : 123,456.789
IN Representation of Currency : ₹123,456.79
US Representation of Currency : $123,456.79
UK Representation of Currency : £123,456.79
IN Representation of Percent : 12,345,679%
US Representation of Percent : 12,345,679%
UK Representation of Percent : 12,345,679%

Process finished with exit code 0
```

Setting Maximum and Minimum Fraction and Integer Digits

NumberFormat class defines following methods for this purpose:

1. public void setMaximumFractionDigits(int digit);

2. public void setMinimumFractionDigits(int digit);
3. public void setMaximumIntegerDigits(int digit);
4. public void setMinimumIntegerDigits(int digit);

In number 123456.789, 123456 is Integer Digits and 789 is a Fractional Digits.

Main.java

Run

```
import java.util.*;
import java.text.*;

class NumberFormatDemo {
    public static void main(String[] args) {
        NumberFormat nf = NumberFormat.getInstance();
        nf.setMaximumFractionDigits(2);
        System.out.println(nf.format(123.4567));
        System.out.println(nf.format(123.4));

        nf.setMinimumFractionDigits(2);
        System.out.println(nf.format(123.4567));
        System.out.println(nf.format(123.4));

        nf.setMaximumIntegerDigits(3);
        System.out.println(nf.format(123456.789));
        System.out.println(nf.format(1.2345));

        nf.setMinimumIntegerDigits(3);
        System.out.println(nf.format(123456.789));
        System.out.println(nf.format(1.2345));
    }
}
```

Output

```
java -cp /tmp/ir4Hw2hHni NumberFormatDemo
123.46
123.4
123.46
123.40
456.79
1.23
456.79
001.23
```

DateFormat

- Various Locations follow various styles to represent a Dates.
  Example:
  IN – 10/08/2019 – DD/MM/YYYY – Read as 10 August, 2019
  US – 10/08/2019 – MM/DD/YYYY – Read as October 8, 2019

- We can use DateFormat class to format or represent the same Date differently as per the Locales.
- DateFormat class is present in java.text package and it is an abstract class.
- We cannot create an Object of DateFormat Class directly. i.e.,
  DateFormat df = new DateFormat(); is not possible in Java.

Getting DateFormat Object for Default Locale

DateFormat Class defines the following methods to get Object for Default Locale:

1. public static DateFormat getInstance();
2. public static DateFormat getDateInstance();
3. public static DateFormat getDateInstance(int style);

Both 1 and 2 are exactly same.

Getting DateFormat Object for Specific Locale

DateFormat Class defines the following methods to get Object for Specific Locales which are same as above three methods, only difference is we have to pass the Object of Locale as an argument for that specific Locales:

1. public static DateFormat getInstance(Locale l);
2. public static DateFormat getDateInstance(Locale l);
3. public static DateFormat getDateInstance(int style, Locale l);

The Values of int styles can be:

0. DateFormat.FULL - e.g. Monday 12th June 2023
1. DateFormat.LONG – e.g. 12th June 2023
2. DateFormat.MEDIUM – e.g. 12th Jun 2023
3. DateFormat.SHORT – e.g. 12/06/23

The default style is 2 (DateFormat.MEDIUM).

Once we get DateFormat Object we can call format() or parse() methods on that Object.

1. public String format(Date d);

This method convert Java Date form to Locale specific String form.

2. public Date parse(String s) throws ParseException;

This method converts Locale specific String form to Java Date form.

Sample Java Program to demonstrate the DateFormat Class Use

Main.java

Run

Output

```
import java.util.*;
import java.text.*;

class DateFormatDemo {
    public static void main(String[] args) {
        DateFormat df1 = DateFormat.getDateInstance(0, Locale.US);
        System.out.println(df1.format(new Date()));
        DateFormat df2 = DateFormat.getDateInstance(1, Locale.US);
        System.out.println(df2.format(new Date()));
        DateFormat df3 = DateFormat.getDateInstance(2, Locale.US);
        System.out.println(df3.format(new Date()));
        DateFormat df4 = DateFormat.getDateInstance(3, Locale.US);
        System.out.println(df4.format(new Date()));
    }
}
```

```
java -cp /tmp/ir4Hw2hHni DateFormatDemo
Monday, June 12, 2023
June 12, 2023
Jun 12, 2023
6/12/23
```

Main.java

Run

Output

```
import java.util.*;
import java.text.*;

class DateFormatDemo {
    public static void main(String[] args) {
        Locale l = new Locale("en","IN");
        DateFormat df1 = DateFormat.getDateInstance(0, l);
        System.out.println(df1.format(new Date()));
        DateFormat df2 = DateFormat.getDateInstance(1, l);
        System.out.println(df2.format(new Date()));
        DateFormat df3 = DateFormat.getDateInstance(2, l);
        System.out.println(df3.format(new Date()));
        DateFormat df4 = DateFormat.getDateInstance(3, l);
        System.out.println(df4.format(new Date()));
    }
}
```

```
java -cp /tmp/ir4Hw2hHni DateFormatDemo
Monday, 12 June, 2023
12 June 2023
12-Jun-2023
12/06/23
```

Note: Apart from Date if we want time also we can use getDateTimeInstance() apart from this everything will be same.

1. public static DateFormat getDateTimeInstance();
2. public static DateFormat getDateTimeInstance(int dateStype, int timeStyle);
3. public static DateFormat getDateTimeInstance(int dateStype, int timeStyle, Locale l);

The allowed styles of time is also 0-3 like date.

Main.java

Run

Output

```
1  import java.util.*;
2  import java.text.*;
3
4  class DateFormatDemo {
5      public static void main(String[] args) {
6          Locale l = new Locale("en","IN");
7          DateFormat df1 = DateFormat.getDateTimeInstance(0,0,l);
8          System.out.println(df1.format(new Date()));
9          DateFormat df2 = DateFormat.getDateTimeInstance(1,1,l);
10         System.out.println(df2.format(new Date()));
11         DateFormat df3 = DateFormat.getDateTimeInstance(2,2,l);
12         System.out.println(df3.format(new Date()));
13         DateFormat df4 = DateFormat.getDateTimeInstance(3,3,l);
14         System.out.println(df4.format(new Date()));
15     }
16 }
```

```
java -cp /tmp/ir4Hw2hHni DateFormatDemo
Monday, 12 June, 2023 at 9:11:23 AM Greenwich Mean Time
12 June 2023 at 9:11:23 AM GMT
12-Jun-2023, 9:11:23 AM
12/06/23, 9:11 AM
```