AP COMPUTER SCIENCE A – SIMPLE INHERITANCE

Consider the `class Date`, listed here.

```
import java.util.Calendar;
/**/
public class Date {
  /**/
  private static final String [] DAY = new String [] {
    "Sunday"   , "Monday" , "Tuesday"  , "Wednesday" ,
    "Thursday" , "Friday" , "Saturday"
  };
  private static final String [] MONTH = new String [] {
    "January"   , "February" , "March"    , "April"    ,
    "May"       , "June"     , "July"     , "August"   ,
    "September" , "October"  , "November" , "December"
  };
  /**/
  private int dow, mnth, dom, year;
  /**/
  public Date () {
    /**/
    init();
  }
  /*
   *  0 ... 6
   */
  public int getDayOfWeek () {
    return dow;
  }
  /*
   *  0 ... 11
   */
  public int getMonth () {
    return mnth;
  }
  /*
   *  1 ... 31
   */
  public int getDayOfMonth () {
    return dom;
  }
  /**/
  public int getYear () {
    return year;
  }
  /**/
  public String toString () {
    /**/
    return DAY[dow]    + " "
         + MONTH[mnth] + " "
         + dom         + ", "
         + year;
  }
  /**/
  private void init () {
    /**/
    Calendar c;
    /**/
    c=Calendar.getInstance();
    dow=c.get(Calendar.DAY_OF_WEEK)-1;
    mnth=c.get(Calendar.MONTH);
    dom=c.get(Calendar.DAY_OF_MONTH);
    year=c.get(Calendar.YEAR);
    /**/
    return;
  }
}
```

For example, the code segment

```
Date d = new Date ();
System.out.println(d);
```

prints out the current date, like

```
Monday November 2, 2020
```

Say we want to print out the date in Spanish. We already have the code which does it in English, so it would be nice just to write a class which uses `Date`, but which just changes parts of it to Spanish (so we don't have to rewrite all of the code). The mechanism in Java which allows for this is called *inheritance*. One of the three cases in which one should use inheritance is when you have a piece of code which does almost what you want, *i.e.*, `Date` almost does what we want.

Consider the `class SpanishDate extends Date` listed here.

```
public class SpanishDate extends Date {
  /**/
  private static final String [] DAY = new String [] {
    "domingo" , "lunes"   , "martes" , "miercoles" ,
    "jueves"  , "viernes" , "sabado"
  };
  private static final String [] MONTH = new String [] {
    "enero"      , "febrero" , "marzo"     , "abril"     ,
    "mayo"       , "junio"   , "julio"     , "agosto"    ,
    "septiembre" , "octubre" , "noviembre" , "diciembre"
  };
  /**/
  public SpanishDate () {
    /**/
    super();
  }
  /**/
  @Override
  public String toString () {
    /**/
    return DAY[getDayOfWeek()] + " "
         + MONTH[getMonth()]   + " "
         + getDayOfMonth()     + ", "
         + getYear();
  }
}
```

Here the code segment

```
SpanishDate d = new SpanishDate ();
System.out.println(d);
```

prints out the current date, in Spanish, like

```
lunes noviembre 2, 2020
```

Let's look at `SpanishDate` more closely.

1) For one thing, `DAY` and `MONTH` have been changed to Spanish.

2) In the constructor, `super()` calls the constructor of class `Date`. `Date` is called the *super-class*. `SpanishDate` is called a *sub-class* of `Date`.

3) A sub-class inherits all of the `public` fields and methods of the super-class. So, `SpanishDate` inherits the methods `getDayOfWeek`, `getMonth`, `getDayOfMonth` and `getYear`. These inherited methods are used in class `SpanishDate` in the `toString()` method. The `@Override` keyword tells Java that this `toString()` method *overrides* the `toString()` method in class `Date`.

4) Finally, a sub-class does not inherit any `private` fields or methods of the super-class. So, `SpanishDate` does not inherit the variables `dow`, `mnth`, `dom`, `year`, `DAY` and `MONTH` in class `Date`.

Now, let's look at the program `TestDate.java` listed here.

```
public class TestDate {
  /**/
  public static void main ( String [] arg ) {
    /**/
    int option;
    Date d;
    /**/
    option=option();
    if ( option == 1 ) d=new Date();
    else               d=new SpanishDate();
    /**/
    System.out.println();
    System.out.println(d);
    /**/
    return;
  }
  /**/
  private static int option () {
    /**/
    int rv;
    ConsoleInput ci;
    /**/
    ci=new ConsoleInput();
    System.out.println();
    System.out.println("Enter option:");
    System.out.println("  English = 1");
    System.out.println("  Spanish = 2");
    while ( true ) {
      rv=ci.readInt("          ? ");
      if ( ( rv == 1 ) || ( rv == 2 ) ) break;
    }
    ci.close();
    /**/
    return rv;
  }
}
```

When the program is run, it asks you if you want the date in English or Spanish, and then it prints out the date in the language chosen. Note that in `main`, the `SpanishDate` object `d` is declared as being a `Date`. In Java, declaring an object to be one of the super-class is called *polymorphism*.

5) Compile and run `TestDate` just to verify that it does what it is supposed to.

6) Write a class `GermanDate extends Date` whose `toString()` method gives the date in German. In German, the days of the week are:

`Sonntag, Montag, Dienstag, Mittwoch, Donnerstag, Freitag,` and `Samstag.`

The months of the year are:

`Januar, Februar, Marz, April, Mai, Juni,`
`Juli, August, September, Oktober, November` and `Dezember.`

AP COMPUTER SCIENCE A – SIMPLE INHERITANCE

7) Write a class `FrenchDate extends Date` whose `toString()` method gives the date in French. In French, the days of the week are:

`dimanche, lundi, mardi, mercredi, jeudi, vendredi` and `samedi.`

The months of the year are

```
janvier, fevrier, mars,      avril,   mai,         juin,
juillet, aout,    septembre, octobre, novembre and decembre.
```

8) Finally, modify program `TestDate.java` into `TestDate2.java`, which can print out the date in either English, Spanish, German or French.

Submit your program files `GermanDate.java`, `FrenchDate.java` and `TestDate2.java` electronically.