EXERCISES WITH PROGRAM Olympics.js

Exercise 1:

Program *Olympics.js* has an initialized array to store objects of type `Olympics`. Update the olympics list of the program so that the latest known Summer Olympic games have `Olympics` objects created. You can also add information about future Summer Olympics.

Ensure that the program works correctly after these modifications.

Exercise 2:

Create a new class named `WinterOlympics` so that the new `WinterOlympics` class will be a subclass of the original `Olympics` class. The first definition of the `WinterOlympics` class could be the following

```
class WinterOlympics extends Olympics
{
   constructor( given_olympic_year,
                given_olympic_city,
                given_olympic_country )
   {
      super( given_olympic_year,
             given_olympic_city,
             given_olympic_country ) ;
   }
}
```

You can put the new `WinterOlympics` class right after the `Olympics` class in the program. With the keyword `extends` we can make `WinterOlympics` to inherit the `Olympics` class. We can say also that class `WinterOlympics` is *derived* from the `Olympics` class. Class `Olympics` is is the superclass of class `WinterOlympics`. With the keyword `super` the constructor of superclass can be called.

When you write the `WinterOlympics` class as shown above, it will behave in the same way as its superclass `Olympics`.

You can test your new class by adding the following object to the array

```
new WinterOlympics( 2006, "Turin", "Italy" )
```

It is possible to have both `Olympics` and `WinterOlympics` objects in the same array. If your program can find the data of Turin olympics, you have successfully carried out this exercise.

Exercise 3:

Improve the new `WinterOlympics` class by writing a new version of method `get_full_data()` into it. The new method should return a text that contains the word 'winter'. The output of the method could look like the following

```
In 2006, Winter Olympics were held in Turin, Italy.
```

In this exercise you can copy the corresponding method from class `Olympics`, and modify the text that is generated by the method.

When a subclass contains a method that has the same name and similar parameters as a method in the superclass, we say that the method is *overridden* in the subclass. In this exercise we override the method `get_full_data()` and the new version of the method will automatically be used for `WinterOlympics` objects.

Exercise 4:

Earlier the winter and summer olympics were orginized during the same year. For example, in 1984 the winter olympics were in Sarajevo, Yogoslavia, and the summer olympics were in Los Angeles, U.S.A.

If you add `WinterOlympics` objects that describe these earlier winter games to the olympics list of the program, there will be problems. The search algorithm finds only the first olympics that were held in the given year. If there are two objects for the same year, the latter object will not be found.

Modify the program so that it will always process all objects of the list of olympics, and print data of those objects that contain the given year.

One way to do this is that you use another array to store those `Olympics` objects that need to be printed. Now the program has the variable

```
var selected_olympics = null ;
```

You can modify this statement so that you specify an empty array in the following way

```
var selected_olympics = [] ;
```

Then you can go through all `Olympics` objects of `olympics_list`, and with the `push()` method add those objects whose year matches the given year to the above array.

After all `Olympics` objects have been checked, you can test the `length` of the above array, and print information about as many `Olympics` as there are objects in the above array.

To test the new version of the program you should add more `WinterOlympics` objects to the initialized array.

Exercise 5:

Now the **Olympics.js** program accepts only numbers as input from the user. Improve the program so that if the user types in the word "summer", the program will print data of all summer olympic games. Then, if the user types "winter", the data of all winter games will be printed.

JavaScript has a built-in global function named `isNaN()` which returns `true` if something is 'not-a-number'. You can use this function to test if the user typed in a number or a word. The function can be called in the following way

```
if ( isNaN( input_from_user ) )
{
   var given_text = String( input_from_user ).trim() ;
   ...
```

As shown above, you should convert the user input to a string and `trim()` it before you start checking what word was typed.

Another you have to solve while doing this exercise is to find out how you decide whether an object in the array is of type `Olympics` or of type `WinterOlympics`. You can solve this problem by using the `instanceof` operator of JavaScript.

The boolean expression in the following `if` construct will be `true` if `olympics_object` refers to an object that is of type `WinterOlympics` or of some subtype of `WinterOlympics`.

```
if ( olympics_object instanceof WinterOlympics )
{
   ...
}
```

Then, if you would like to know whether some olympics object is of type `Olympics`, you can use an `if` construct such as

```
if ( ! ( olympics_object instanceof WinterOlympics ) )
{
   ...
}
```

In this exercise, you cannot use the `instanceof` operator to test whether an object is of type `Olympics`. The reason for this is that `instanceof` returns `true` also when the object on its left side is an object of some subclass of the class given on the right side of the operator.

Exercise 6:

If the user types in a text but the text is not "summer" or "winter", your program should suppose that the user typed in the name of a country or a city.

Make an `else` part to your program so that it searches all `Olympics` or `WinterOlympics` objects that contain the given text. For example, if the user types "U.S.A.", the program should print a list like the following:

```
In 1904, Olympic Games were held in St. Louis, U.S.A..
In 1932, Olympic Games were held in Los Angeles, U.S.A..
In 1984, Olympic Games were held in Los Angeles, U.S.A..
In 1996, Olympic Games were held in Atlanta, U.S.A..
```

Or if the user types "Italy", the program should print something like

```
In 1960, Olympic Games were held in Rome, Italy.
In 2006, Winter Olympics were held in Turin, Italy.
```