Answer on Question# 48936 - Programming – Java

Question:

“1.What is method overloading?

2. What is the purpose of a constructor?

3. How do you call a method of one class from a method of another class?

4.(a.) Explain how Java programs are compiled and run, and how this differs from the way most other programs are compiled and run.

(b.) Explain with the aid of a diagram the advantages of object oriented approach over structured approach to modular design.

(c.) Write a method that (a) returns the maximum value in an array (b) a method that searches an array and returns the position of an item within the array or -999 if the value is not present in the array.

(i). Write a short menu driven program that makes use of the above methods(array method)”.

Answer:

1. Method overloading is a process of definition within the same class of two or more methods with the same names but with the different parameters.

2. The purpose of a constructor is the object initialization directly during its creation. Constructor defines of actions which must be performed within the creating of the class object.

3. It should to use a reference (*ourObject*) for class object which has the method (*ourMethod ()*). Next are the point operator and *ourMethod ()*. Like this:

```
void anotherMethod (){//this is the method of another class
        ourObject.ourMethod ();
}
```

4. (a)Unlike other programs, in Java a result of compiler isn't an executable code. It's a bytecode for Java run-time system (Java Virtual Machine - JVM) which interprets it and runs the program.

   For example:

   C:\>javac Program.java,

   where *javac* is the java compiler, *Program* is the name of source file, *java* is the file extension.

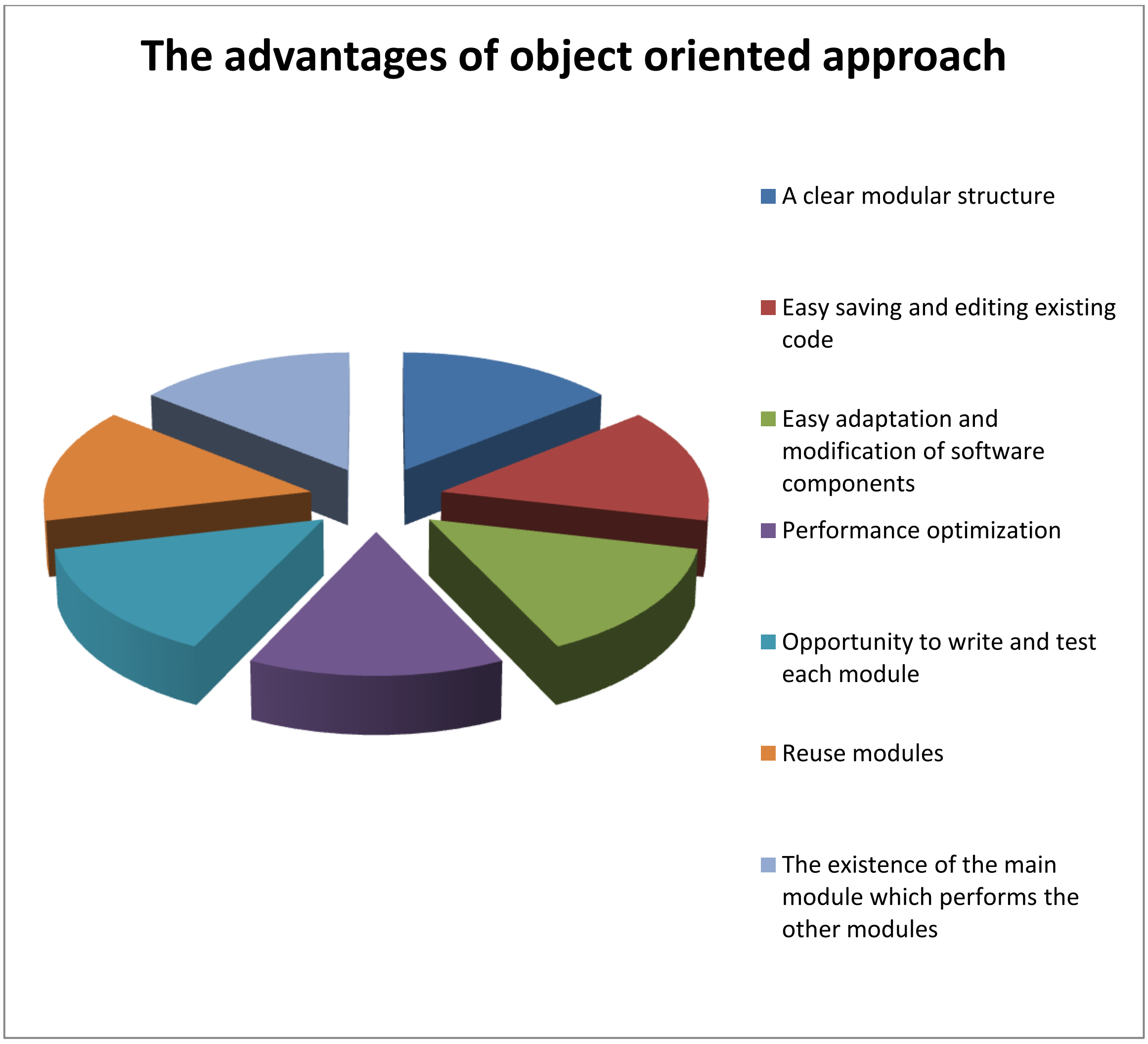
   After the launch of the compiler we have a file *Program.class* with the *.class* extension which contains a JVM bytecode.

   C:\>java Program,

   where *java* is the Application Launcher, *Program* is the name of the file with *.class* extension.

   As we can see, *Java* runs *Program*, i.e. interprets bytecode to executable code. In other programs, there is no *bytecode* as a result of compiler is a directly executable code.

   (b)



(c):

a)

```
int maxValue(int []array){//it takes the array

        int maxValue = 0;//set the initial value of maximum value

        for(int index = 0; index < array.length; index ++){//loop through an array

                if(array[index] > maxValue){// check whether the current element in the array
//more than maxValue

                        maxValue = array[index];// if so, assign the value of the current array
//element to maxValue
                }
        }

        return maxValue;
}
```

b)

```
int elementPosition(int []array, int element){// it takes the array and given element

        for(int index = 0; index < array.length; index ++){//loop through an array

                if(array[index] == element){// check whether the current element
// is equal to a given element


                        return index;//return the index
                }

        }

        return -999;//or return the value -999
}
```

(i)

```
public class My {
```

```java
static int maxValue(int []array){

        int maxValue = 0;

        for(int index = 0; index < array.length; index ++){

                if(array[index] > maxValue){

                        maxValue = array[index];
                }
        }

        return maxValue;
}

static int elementPosition(int []array, int element){

        for(int index = 0; index < array.length; index ++){

                if(array[index] == element){

                        return index;
                }

        }

        return -999;
}

public static void main (String[]args){

        int []array = {1, 19, 3, 6, 18};//create an array

        System.out.println(maxValue(array));//get the maximum value

        System.out.println(elementPosition(array, 3));//get the position of element "3"

        System.out.println(elementPosition(array, 5));//try to get the position of element "5"
//(this element isn't present in the array)
}
}
```

Execution output:

```
19
2
-999
```