

15/May/2008

V0.6

1. General Programming principles - just an overview on programming and coding

General Programming principles

This article will give the reader some insight about the general programming principles,

Learning how to code is the first step to become a fully fledged software developer but being able to code doesn't mean that you can write clean and maintainable code.

The fact is that there are many ways to code an application that can perform the same functionality, such as an algorithm to search and retrieve data from a text file or graphics on the screen. The code that is written by you doesn't always mean that the code can be read or understandable by someone else. You may know what your code does but other developers will not if your code is either:

- *not formatted*, it's very important to format your code for readability. Use tabs and enters where needed but don't clutter your code into one long text of code or use unnecessary tabs and enters.

Example #1 (bad):

```
public class HelloWorld {public static void main(String[] args)
{System.out.println("General Programming principles article");}}
```

Example #2 (bad):

```
public class HelloWorld
    public static void main(String[] args)        {

        System.out.println("General Programming principles article");
    }

}
```

Example (good):

```
public class HelloWorld {
    public static void main(String[] args){
        System.out.println("General Programming principles
article");
    }
}
```

Notice the long text of code in example #1 and the massive use of tabs and enters in example #2

- self explaining object names, do you know what I mean by:

```

public class HelloWorld {
    String tmp = "dog";
    public static void main(String[] args){
        HelloWorld HW = new HelloWorld();
        HW.setTheString("cat");
        System.out.println(HW.getTheString());
    }

    public void setTheString(String tmp){
        this.tmp = tmp;
    }

    public String getTheString(){
        return tmp;
    }
}

```

Notice this: String tmp = "dog"; ? setTheString(String tmp) ? , getTheString() ?

I have no idea what this application should do by judging the object names for this application. But ok, just to be fair I can see that the String tmp holds the value "dog" and the getTheString(String tmp) will do something with a String?

Wouldn't be easier to read if the names were changed into:

```

public class Animal {
    String AnimalStr = "dog";
    public static void main(String[] args){
        Animal animalObject = new Animal();
        System.out.println(animalObject.getTheAnimal());
        animalObject.setTheAnimal("cat");
        System.out.println(animalObject.getTheAnimal());
    }

    public void setTheAnimal(String Animal){
        this.AnimalStr = Animal;
    }

    public String getTheAnimal(){
        return AnimalStr;
    }
}

```

With a little refactoring I can see that this is an Animal class that sets and gets an animal. Imagine how the code would look like in a real world application with thousands and thousands lines of code with object names like: tmp, String2, intNumber, aDouble, useThisString etc. Well, I would go to the programmer that programmed this and ask for explanation because he has a lot of explaining to do.

- *comments* , when writing large blocks of code comment it!

Even the best developer won't know what the code will do after months of not touching the code. Without comments to clarify what the code does the developer will easily lose track what it's supposed to do and spent more time to figure out what it's supposed to do than necessary..

When I write large blocks of code I write comments what the method is doing above the methods that I start creating like this:

```
// set the animal
public void setTheAnimal(String Animal){
    this.AnimalStr = Animal;
}
```

or if you need multiple lines:

```
/*
 * set the animal
 */
public void setTheAnimal(String Animal){
    this.AnimalStr = Animal;
}
```

Of course there isn't always a need to comment a method if it's already obvious what it will do like my example above.

Understanding the problem

Another important thing to remember is the ability to fully understand the problem. You might think that's it's better to code and see what the code does but you can't just code something without fully understanding the specs of the application because if you don't know what you are doing how are you supposed to write software that does what it needs to do?

What I usually do is to write down the problem step by step on a piece of paper and try to figure out how to solve the problem without a computer. After I've managed to write down the steps it's only a matter of converting the steps into Java or any other programming language.

Conclusion

While there are many ways a developer can code software, there are guidelines to remember so that it makes the code easier to read. Another important thing is to understand the basic OO concepts that I haven't discussed about it yet but understanding the OO concept is crucial in java as everything can be assumed as an object and I'll explain this in another article why OO design is important.

Ark

www.engineeringserver.com