Прекрасный мир Java

Часть 5.

GitHub

https://github.com/amakedonsky/java_lessons

Почему ООП?

Почему Java?

C++:

```
#include "string"
#include "iostream"
#include "windows.h"
using namespace std;
int main(int argc, char* argv[])
{
   long lTime = GetTickCount();
   string test("Hello world!");
   for (int iIdx = 0; iIdx <
   1000000; iIdx++)
   {
      test.substr(1, 10);
   }
   lTime = GetTickCount() -
   lTime;
   cout << lTime << endl;
   return 0;
}
```

Java:

```
public class Test
{
public static void main(String []
args) {
   long lTime =
   System.currentTimeMillis();
   String strTest = "Hello
   world!";
   for (int iIdx = 0; iIdx <
   1000000; iIdx++) {
      strTest.substring(1, 11);
   }
   lTime =
   System.currentTimeMillis() -
   lTime;
   System.out.println(""+lTime);
}
}
```

Строки в Java построены с использованием шаблона «неизменный объект» (**Immutable object**).

Строки в Java построены с использованием шаблона «неизменный объект» (**Immutable object**).

Суть этого шаблона в том, что объект, построенный по этому шаблону, не содержит методов, позволяющих изменить его состояние (а прямое изменение членов объекта невозможно в силу применения инкапсуляции). Получается, что объект, будучи созданным, в дальнейшем не может быть изменен.

Преимущества подхода:

1. При передаче неизменного объекта как параметра в подпрограмму нет необходимости создавать его копию, чтобы предотвратить объект от возможных его модификаций в подпрограмме.

2. Если неизменный объект хранит данные в каких-то внутренних структурах (например буфер в классе String), то эти внутренние структуры можно разделять между несколькими экземплярами объектов класса (например, при взятии подстроки объекты класса String совместно используют один буфер).

Как манипулировать со строками в Java

Как манипулировать со строками в Java

```
StringBuffer strBuffer = new StringBuffer(20);
strBuffer.append("Hello, ");
strBuffer.append("World!");
String strMessage = strBuffer.toString();
```

Как манипулировать со строками в Java

```
StringBuffer strBuffer = new StringBuffer(20);
strBuffer.append("Hello, ");
strBuffer.append("World!");
String strMessage = strBuffer.toString();

String strMsg1 = "Hello, ";
String strMsg2 = "World!";
String strMessage = strMsg1 + strMsg2;
```

Основные понятия ООП

- Инкапсуляция
- Абстракция
  - Интерфейс
- **Наследование**
- **Полиморфизм**

```
class Parent {
    private int iValue;
    Parent(int value) {
        this.iValue = value;
    }
    public int getValue() {
        return this.iValue
    }
}

class Child extends Parent {
    private int newValue;
    Child(int valueOne, int valueTwo) {
        super(valueOne);
        this.newValue = valueTwo;
    }
    public int getAdd() {
        return this.newValue + this.getValue();
    }
}
```

Parent
Int iValue;
Int getValue();

Child
Int iValue;
Int newValue;
Int getValue();
Int getAdd();

Класс, от которого наследуется функциональность, называется **базовым** классом или **суперклассом**.

Класс, который наследует функциональность, называется **наследником**, **подклассом**, или **потомком** соответствующего суперкласса.

Наследование означает получение подклассом всей функциональности суперкласса. Это означает, что фактически наследуются и функциональность класса (закрытый интерфейс), и внешний интерфейс.

Если класс сделал какую-то функциональность видимой извне, то и все потомки обязаны предоставлять эту функциональность.

Методы класса **Object**:

- public String toString()
- public boolean equals(Object obj)
- public Class getClass()
- public int hashCode()
- public final native void notify()
- public final native void notifyAll()
- public final void wait()
- protected Object clone()
- protected void finalize()

Определить метод toString() для класса ComplexNumber