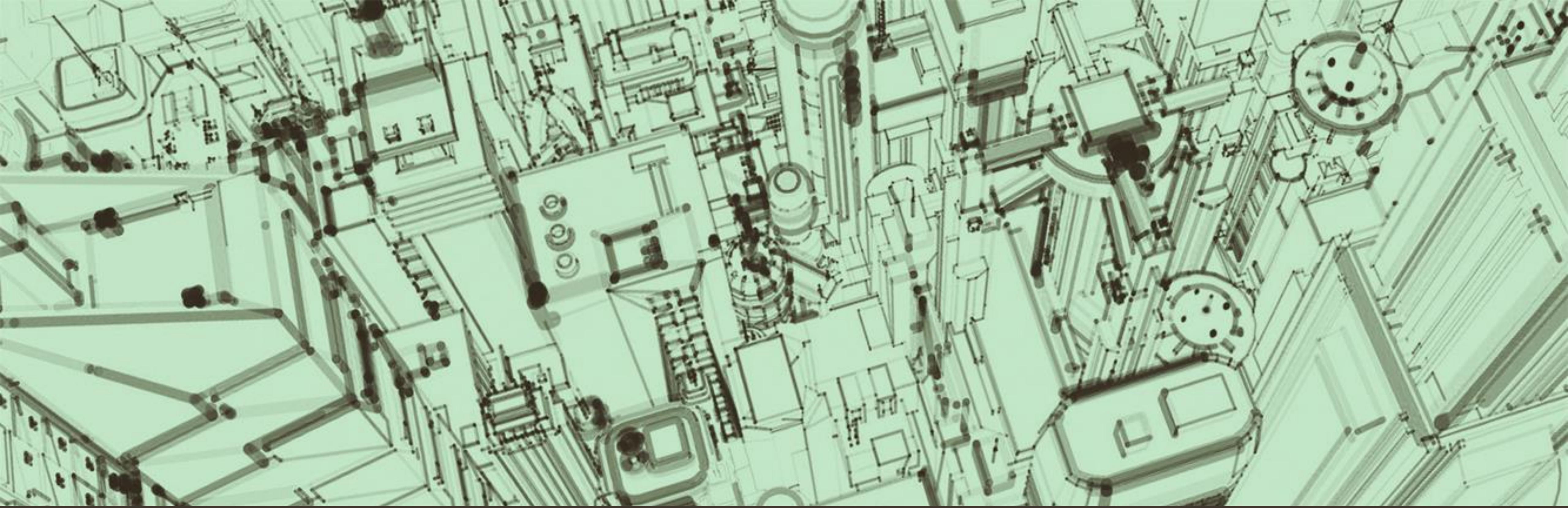
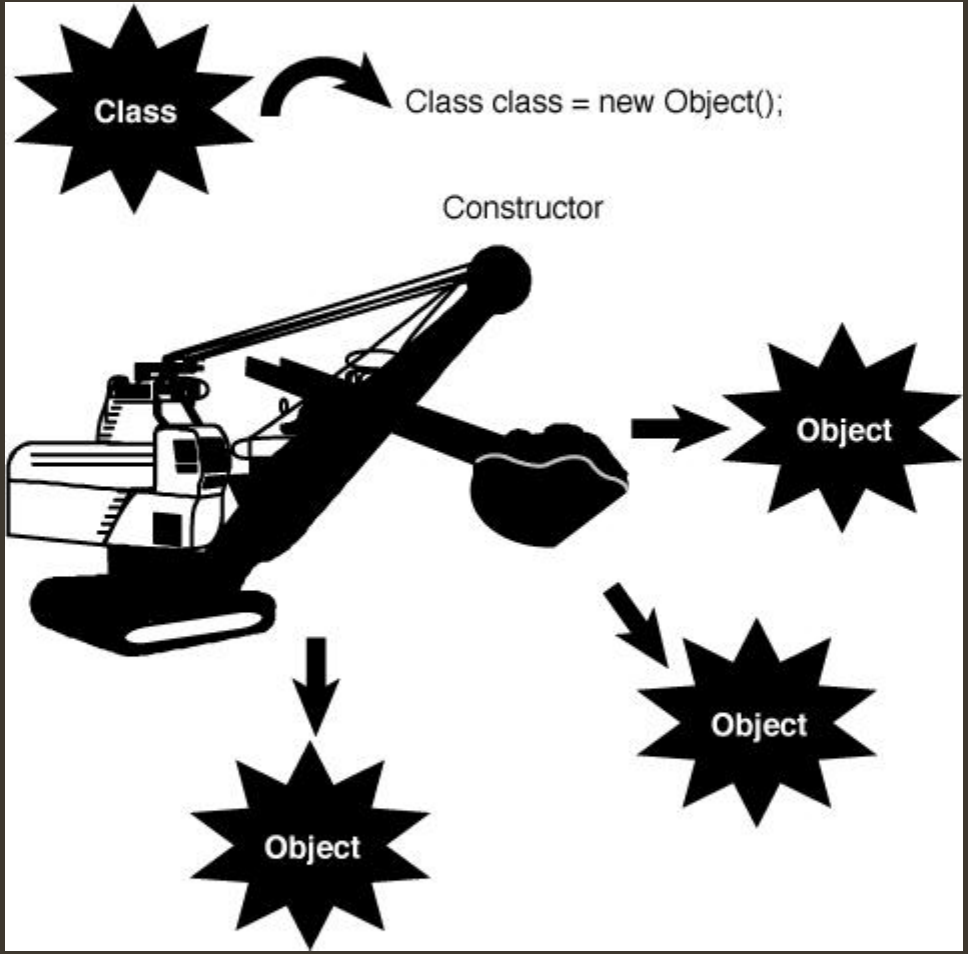


BUILDER DESIGN PATTERN

JAVA


Class
Class class = new Object();
Constructor
Object
Object
Object

Why?

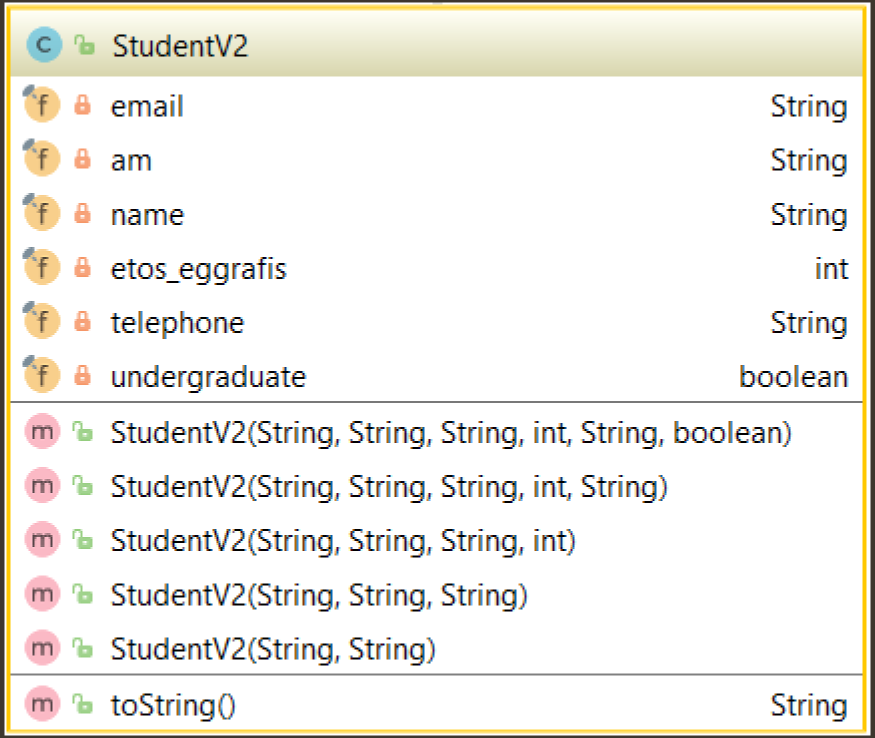
- Creating Objects is one of the most fundamental operations in OO programming
- Each Object is usually (almost always) created through a constructor (directly or sometimes even indirectly)
- In this presentation we are examining the creation of common objects that include a significant number of fields
- There are 3 basic “ways” to do such an object creation:
  - Field initialization in during object construction
  - Field initialization after object construction
  - Field initialization before object construction
- We will also examine advantages and disadvantages in using each design pattern approach!

3 Basic Java Implementations

1. Telescoping Constructor Design Pattern
2. JavaBeans Design Pattern
3. Builder Design Pattern

Telescoping Constructor Design Pattern

```java
package com.unipi.talepis;
public class StudentV2 {
    private final String email; //required
    private final String am; //required
    private final String name;
    private final int etos_eggrafis;
    private final String telephone;
    private final boolean undergraduate;
    public StudentV2(String email,String am,String name,int etos_eggrafis,String telephone,boolean undergraduate){
       this.email = email;
       this.am = am;
       this.name = name;
       this.etos_eggrafis = etos_eggrafis;
       this.telephone = telephone;
       this.undergraduate = undergraduate;
    }
    public StudentV2(String email,String am,String name,int etos_eggrafis,String telephone){
        this(email,am,name,etos_eggrafis,telephone,true);
    }
    public StudentV2(String email, String am, String name, int etos_eggrafis){
        this(email,am,name,etos_eggrafis,"12345",true);
    }
    public StudentV2(String email,String am,String name){
        this(email,am,name,2000,"12345",true);
    }
    public StudentV2(String email,String am){
        this(email,am,"No-Name",2000,"12345",true);
    }
}
```



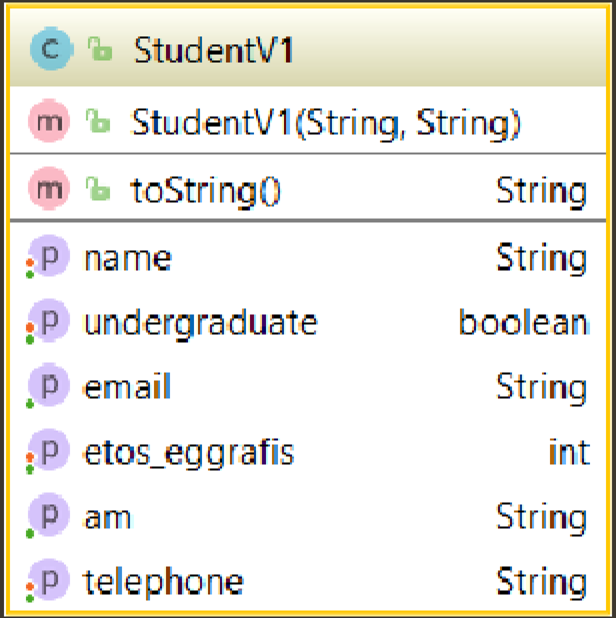
```
StudentV2
email                                            String
am                                               String
name                                             String
etos_eggrafis                                       int
telephone                                        String
undergraduate                                   boolean
StudentV2(String, String, String, int, String, boolean)
StudentV2(String, String, String, int, String)
StudentV2(String, String, String, int)
StudentV2(String, String, String)
StudentV2(String, String)
toString()                                       String
```

Pros and Cons

- Quite “Straightforward” way of object creation (usually it is common sense)
- “Fast” object creation
- Supports “private final” fields!
- Difficult to write code when the number of fields increases
- Does not support all combinations for fields

JavaBeans Design Pattern

```java
package com.unipi.talepis;
public class StudentV1 {
    public StudentV1(String email, String am) {
        this.email = email;
        this.am = am;
    }
    public String getEmail() {
        return email;
    }
    public String getAm() {
        return am;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getEtos_eggrafis() {
        return etos_eggrafis;
    }
    public void setEtos_eggrafis(int etos_eggrafis) {
        this.etos_eggrafis = etos_eggrafis;
    }
    public String getTelephone() {
        return telephone;
    }
    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }
    public boolean isUndergraduate() {
        return undergraduate;
    }
    public void setUndergraduate(boolean undergraduate) {
        this.undergraduate = undergraduate;
    }
    private final String email; //required
    private final String am; //required
    private String name;
    private int etos_eggrafis;
    private String telephone;
    private boolean undergraduate;
}
```



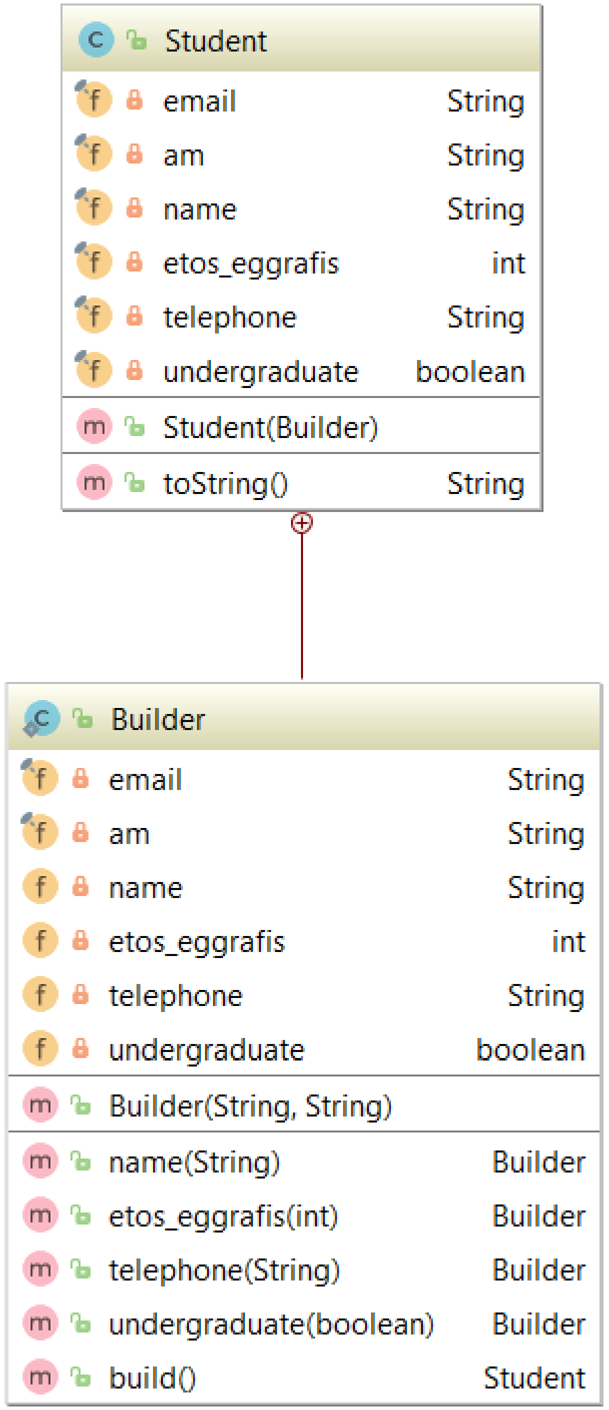
```
StudentV1
StudentV1(String, String)
toString()            String
name                  String
undergraduate         boolean
email                 String
etos_eggrafis         int
am                    String
telephone             String
```

Pros and Cons

- Supports each possible combination of object fields
- Access to fields after object creation (however in many cases this is very dangerous!)
- Easy to create getters and setters through Java IDEs
- Easy to scale - Adding more and more parameters is still easy
- Needs a big number of set commands, when the number of fields increases
- Fields are not final (…)
- Concurrency problems

Builder Design Pattern

```java
package com.unipi.talepis;
public class Student {
    private final String email; //required
    private final String am; //required
    private final String name;
    private final int etos_eggrafis;
    private final String telephone;
    private final boolean undergraduate;
    public Student(Builder builder){
        this.email = builder.email;
        this.am = builder.am;
        this.name = builder.name;
        this.etos_eggrafis = builder.etos_eggrafis;
        this.telephone = builder.telephone;
        this.undergraduate = builder.undergraduate;
    }
    public static class Builder{
        private final String email; //required
        private final String am; //required
        private String name;
        private int etos_eggrafis;
        private String telephone;
        private boolean undergraduate;
        public Builder(String email,String am){
            this.email = email;
            this.am = am;
        }
        public Builder name(String name){
            this.name = name;
            return this;
        }
        public Builder etos_eggrafis(int etos_eggrafis){
            this.etos_eggrafis = etos_eggrafis;
            return this;
        }
        public Builder telephone(String telephone){
            this.telephone = telephone;
            return this;
        }
        public Builder undergraduate(boolean undergraduate){
            this.undergraduate = undergraduate;
            return this;
        }
        public Student build(){
            return new Student(this);
        }
    }
}
```



Student
email String
am String
name String
etos_eggrafis int
telephone String
undergraduate boolean
Student(Builder)
toString() String
Builder
email String
am String
name String
etos_eggrafis int
telephone String
undergraduate boolean
Builder(String, String)
name(String) Builder
etos_eggrafis(int) Builder
telephone(String) Builder
undergraduate(boolean) Builder
build() Student

Pros and Cons

- More “sophisticated” object creation pattern
- A supplementary inner, static, class is utilized
- Fields remain private and final
- Very readable and easy to maintain code!
- A little bit more slow than the telescopic version
- Not very “handy” when objects possess a small number of fields

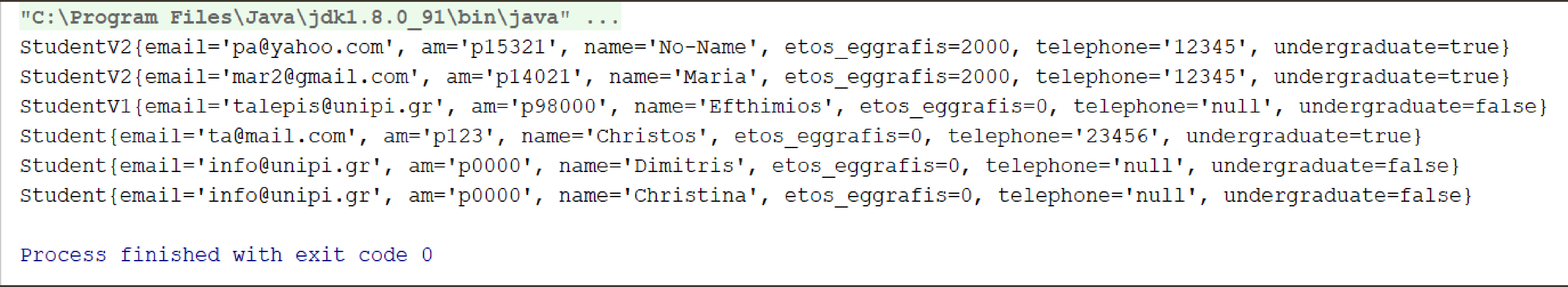
Lets Test them!

```java
package com.unipi.talepis;
public class Main {
    public static void main(String[] args) {
        StudentV2 s1 = new StudentV2("pa@yahoo.com","p15321");
        StudentV2 s2 = new StudentV2("mar2@gmail.com","p14021","Maria");

        StudentV1 s3 = new StudentV1("talepis@unipi.gr","p98000");
        s3.setName("Efthimios");

        Student s4 = new Student.Builder("ta@mail.com","p123")
                .name("Christos")
                .telephone("23456")
                .undergraduate(true)
                .build();
        Student.Builder b1 = new Student.Builder("info@unipi.gr","p0000");
        Student s5 = b1.name("Dimitris").build();
        Student s6 = b1.name("Christina").build();

        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
        System.out.println(s4);
        System.out.println(s5);
        System.out.println(s6);
    }
}
```



```
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ...
StudentV2{email='pa@yahoo.com', am='p15321', name='No-Name', etos_eggrafis=2000, telephone='12345', undergraduate=true}
StudentV2{email='mar2@gmail.com', am='p14021', name='Maria', etos_eggrafis=2000, telephone='12345', undergraduate=true}
StudentV1{email='talepis@unipi.gr', am='p98000', name='Efthimios', etos_eggrafis=0, telephone='null', undergraduate=false}
Student{email='ta@mail.com', am='p123', name='Christos', etos_eggrafis=0, telephone='23456', undergraduate=true}
Student{email='info@unipi.gr', am='p0000', name='Dimitris', etos_eggrafis=0, telephone='null', undergraduate=false}
Student{email='info@unipi.gr', am='p0000', name='Christina', etos_eggrafis=0, telephone='null', undergraduate=false}

Process finished with exit code 0
```