Object Instantiation and Initialization in Java

CSCI 111

Computer Memory

Driver Java Code

What Happens?

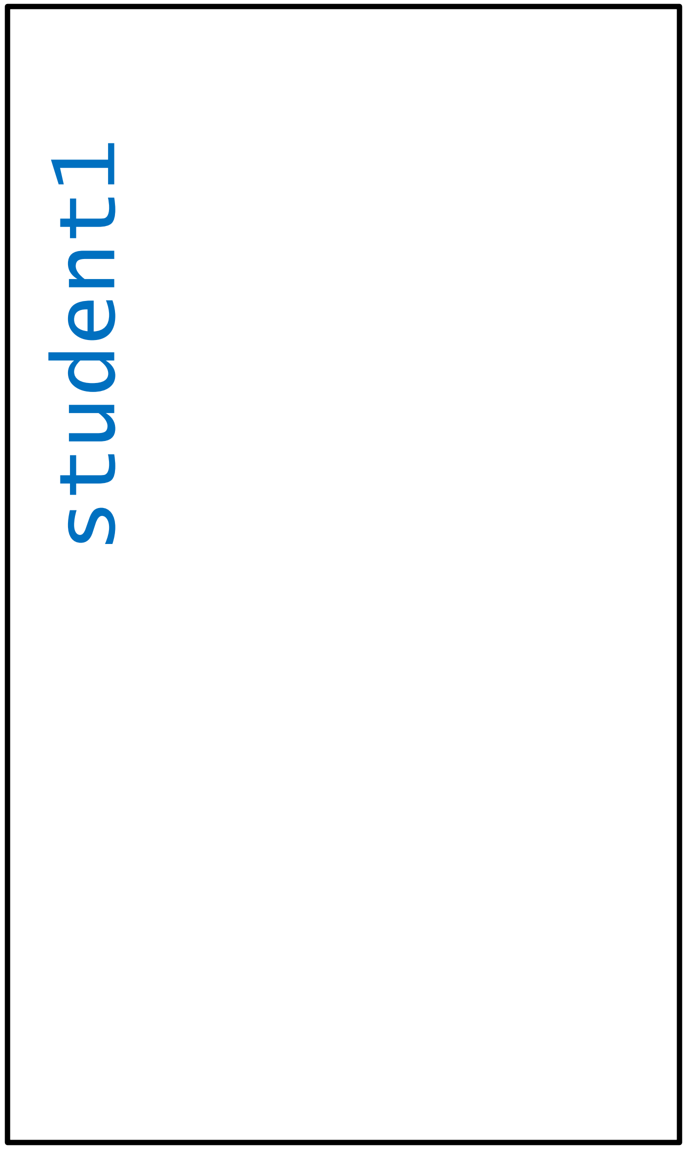
Driver Java Code

```
Student student1;
```

What Happens?

A new variable, `student1`, is created. `student1` can ONLY hold an instance of the Student class. i.e. `student1` cannot hold a String ("Joe") or an integer (6). This is called Variable Declaration.

Computer Memory

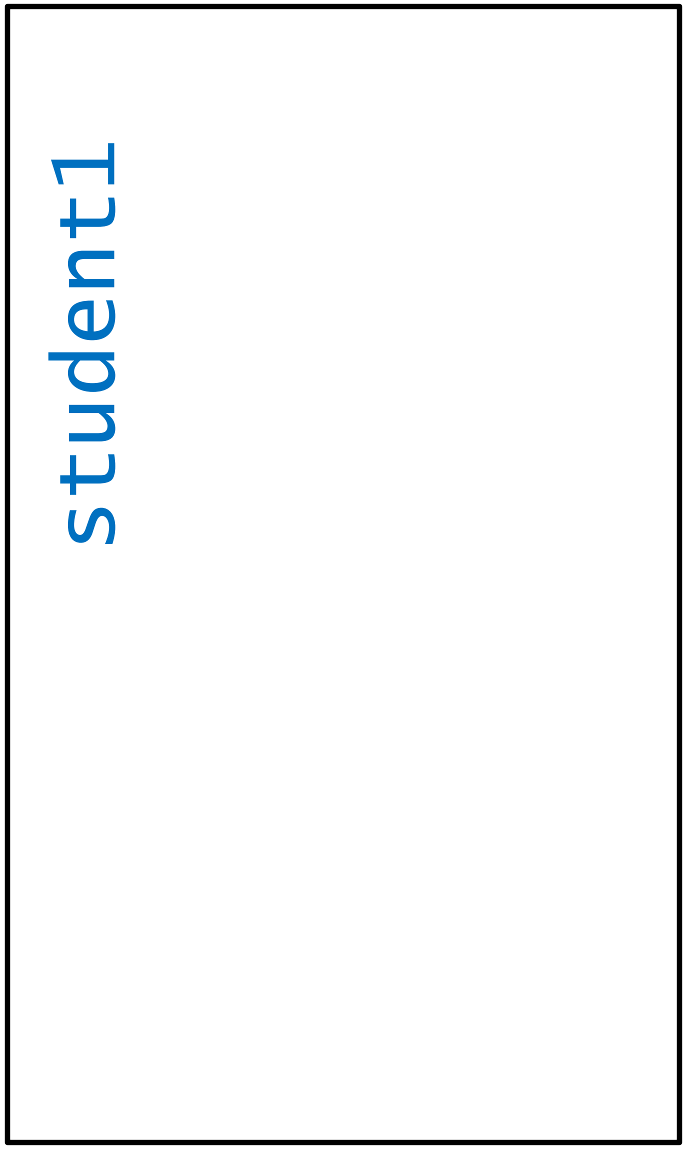


Driver Java Code

```
Student student1;
student1 = new Student(“Joe”, 123, 3.2);
```

What Happens?

Computer Memory



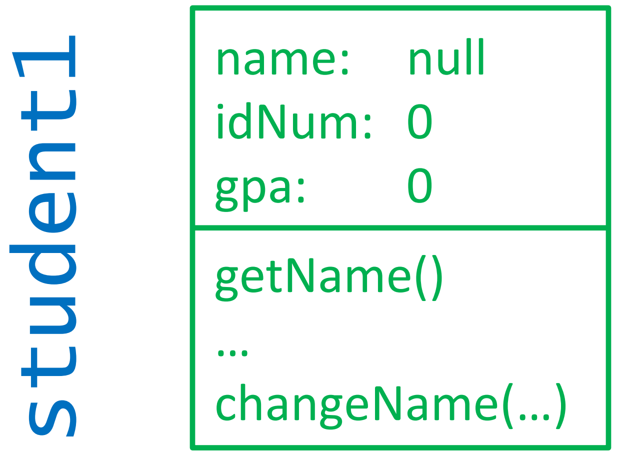
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
```

What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called <u>Instantiation</u>.

Computer Memory



name: null
idNum: 0
gpa: 0

getName()
…
changeName(…)

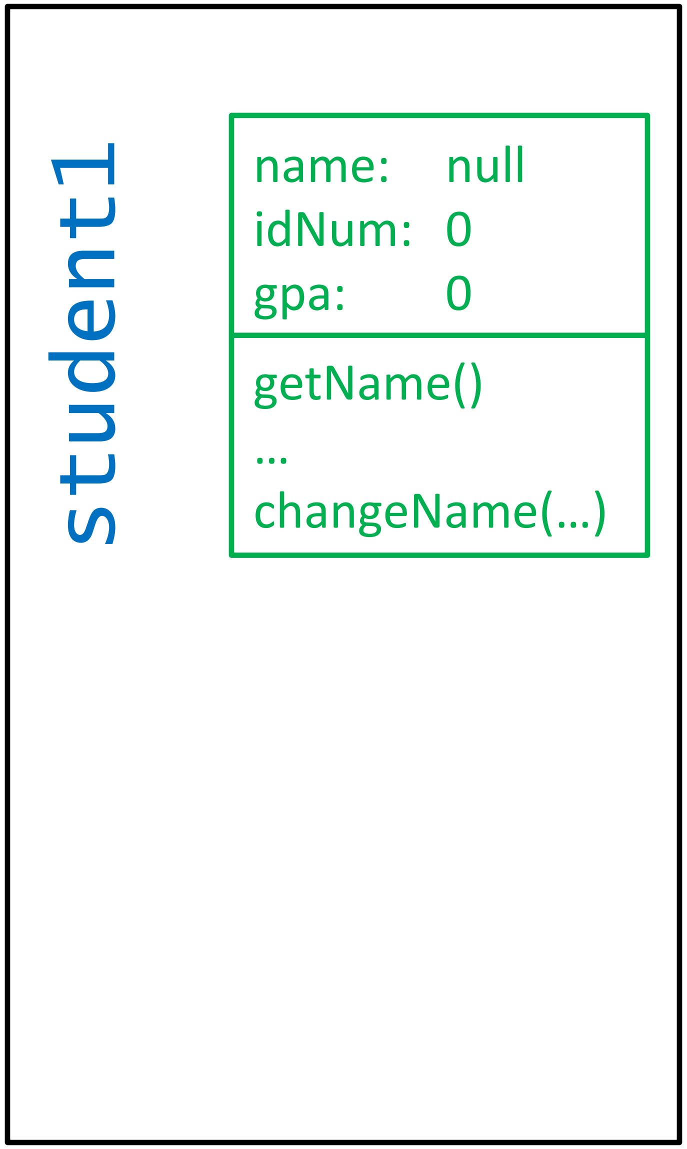
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
```

What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called <u>Instantiation</u>.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called <u>Initialization</u>.

Computer Memory



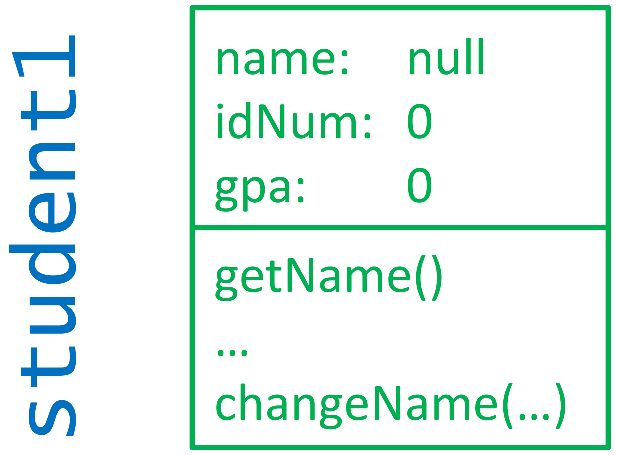
Student Java Code

```
public Student(String inName, int inID, double iG)
{
    name = inName;
    idNum = inID;
    gpa = iG;
}
```

What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called Instantiation.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called Initialization.

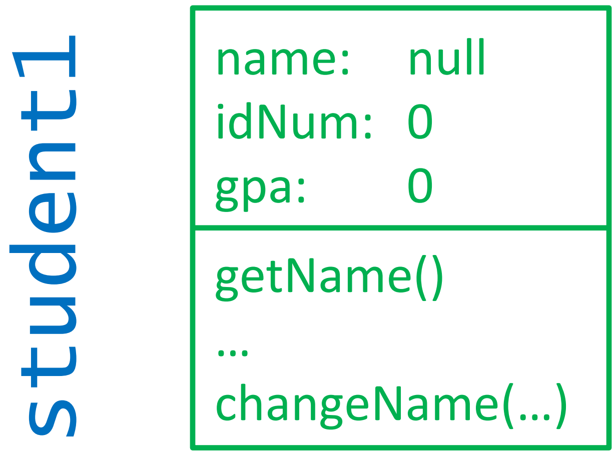
Computer Memory



Student Java Code

```
public Student(~~String inName~~, ~~int inID~~, ~~double iG~~)
{                      “Joe”          123         3.2
    name = inName;
    idNum = inID;
    gpa = iG;
}
```

Computer Memory



What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called Instantiation.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called Initialization.

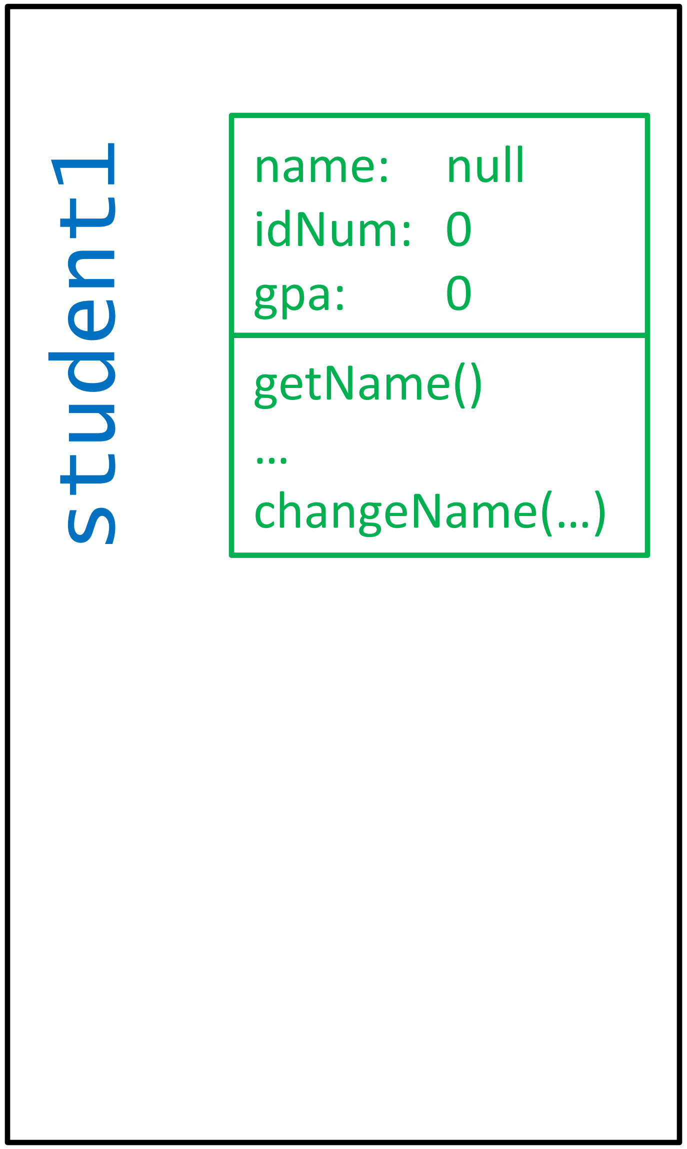
Student Java Code

```
public Student(~~String inName~~, ~~int inID~~, ~~double iG~~)
{                   "Joe"            123           3.2
    name = ~~inName~~; "Joe"
    idNum = ~~inID~~; 123
    gpa = ~~iG~~;3.2
}
```

What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called Instantiation.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called Initialization.

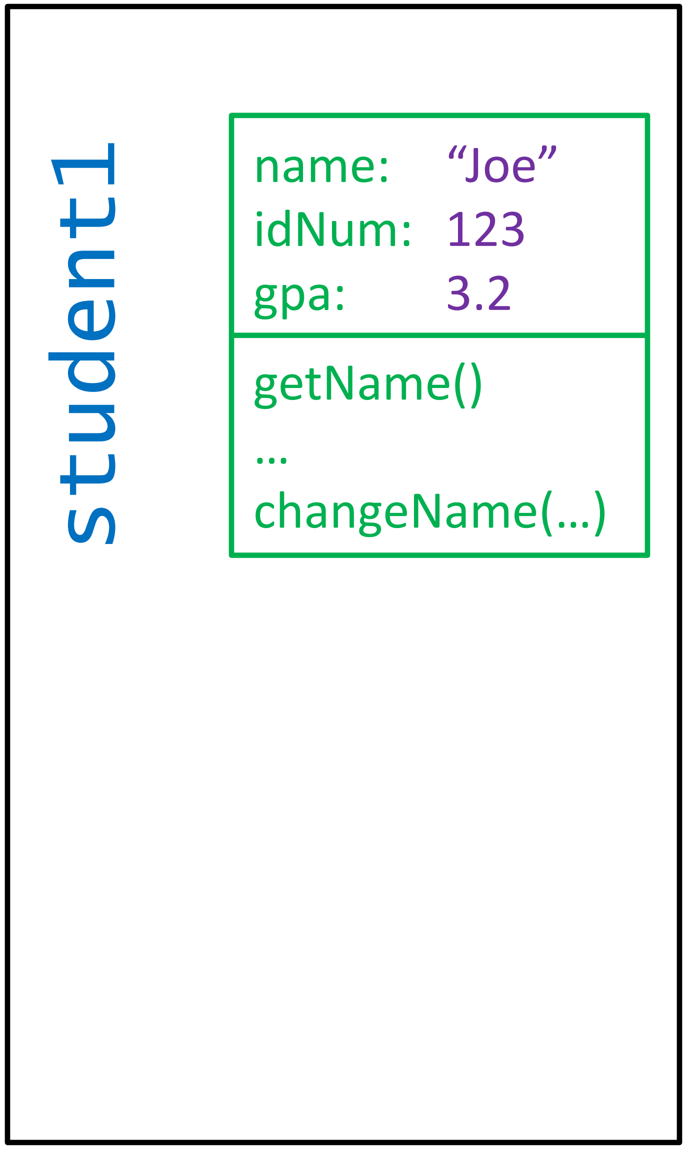
Computer Memory



Student Java Code

```
public Student(~~String inName~~, ~~int inID~~, ~~double iG~~)
{                        “Joe”            123          3.2
    name = ~~inName~~;“Joe”
    idNum = ~~inID~~; 123
    gpa = ~~iG~~;3.2
}
```

Computer Memory



What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called <u>Instantiation</u>.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called <u>Initialization</u>.

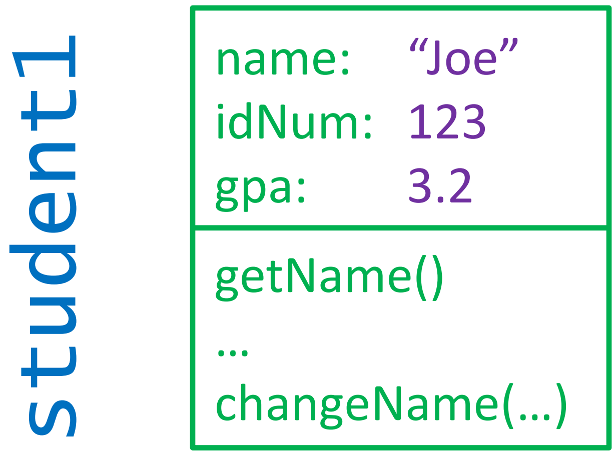
Driver Java Code

```
Student student1;
student1 = new Student(“Joe”, 123, 3.2);
```

What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called Instantiation.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called Initialization.

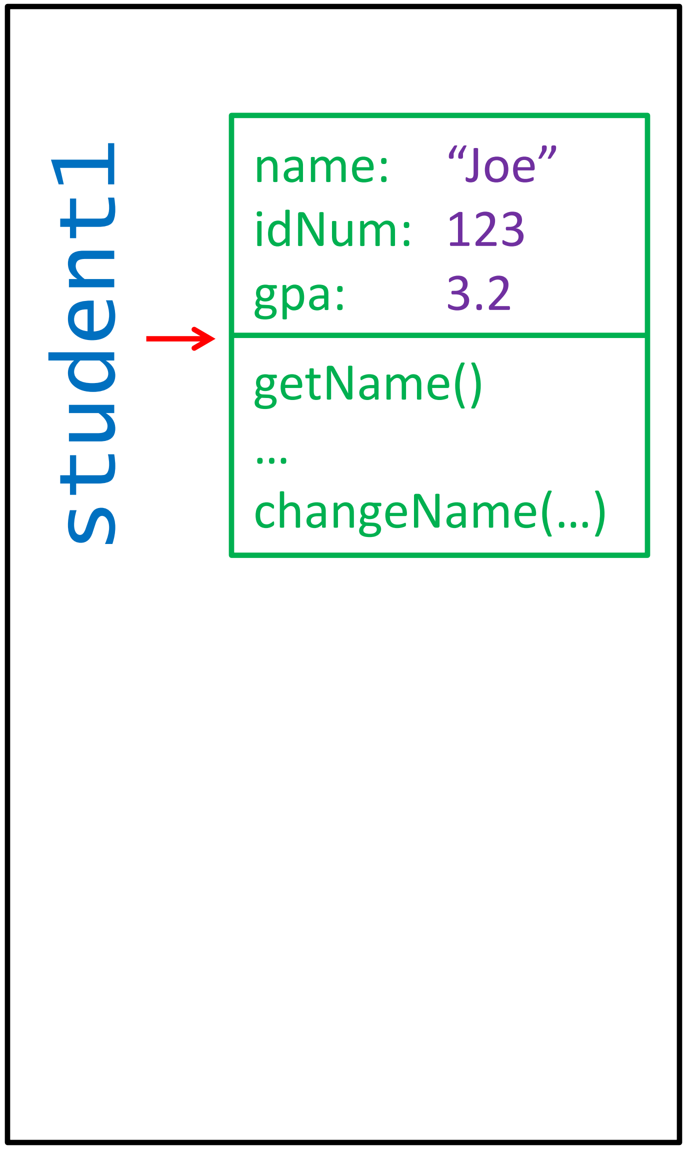
Computer Memory



Driver Java Code

```
Student student1;
student1 = new Student(“Joe”, 123, 3.2);
```

Computer Memory



What Happens?

1. A new Student object (instance of the Student class) is created in memory with the default instance variable values. This set is called <u>Instantiation</u>.
2. The constructor in the Student class is called to populate variables with their initial values. This step is called <u>Initialization</u>.
3. `student1` is set to point to this new object. This step is called <u>Variable Assignment</u>.

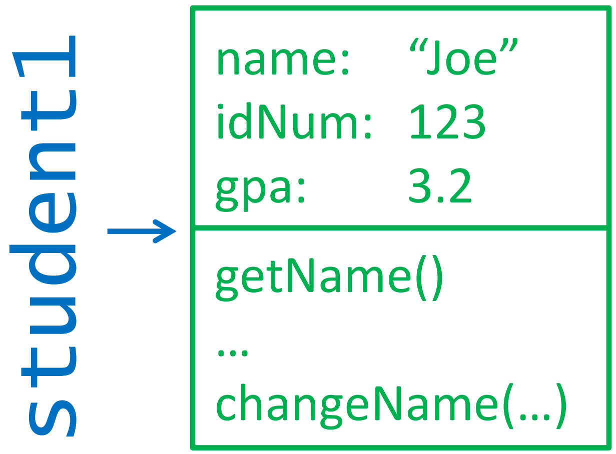
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
```

What Happens?

Whose `getName()` method are we calling?

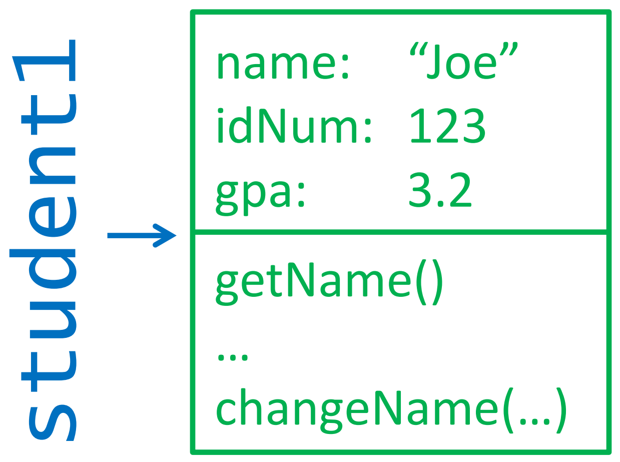
Computer Memory



Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
```

Computer Memory



What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
```

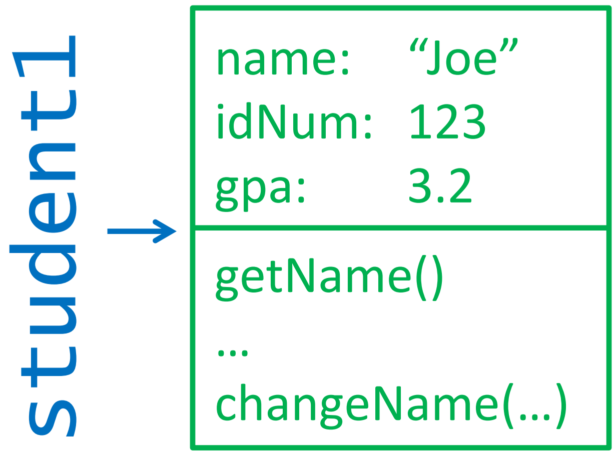
What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

So go to the object that `student1` is pointing to and look at the `getName()` method.

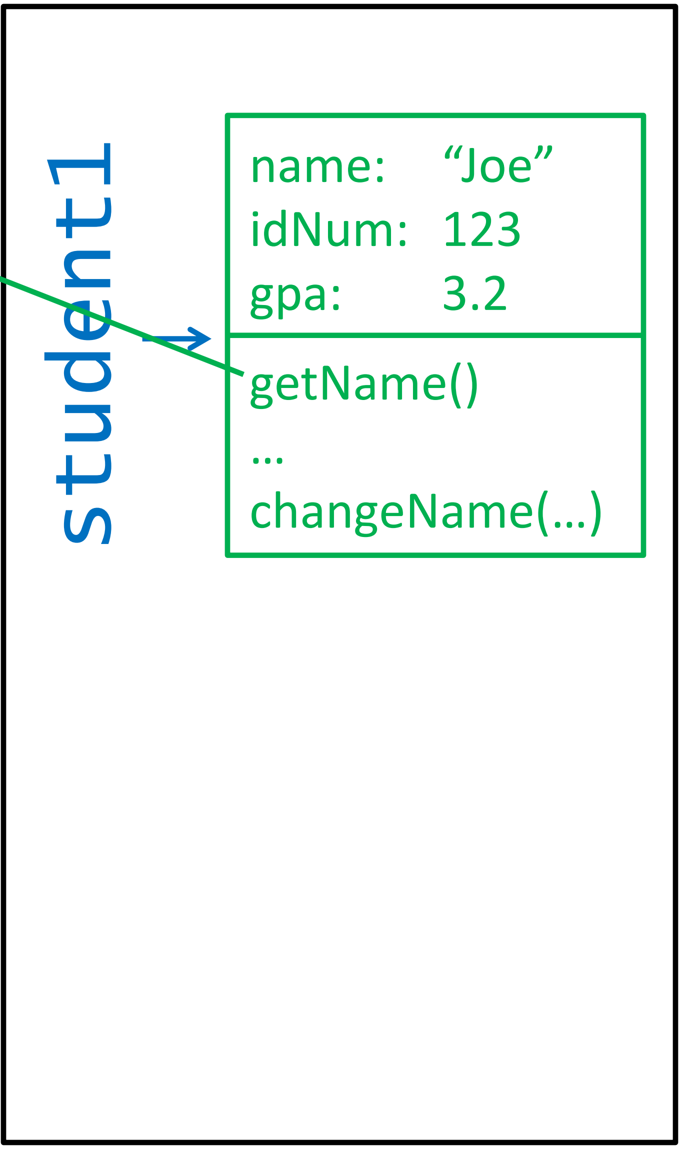
Computer Memory



Student Java Code

```
public String getName()
{
    return name;
}
```

Computer Memory



What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

So go to the object that `student1` is pointing to and look at the `getName()` method.

Student Java Code

```
public String getName()
{
    return name;
}
```

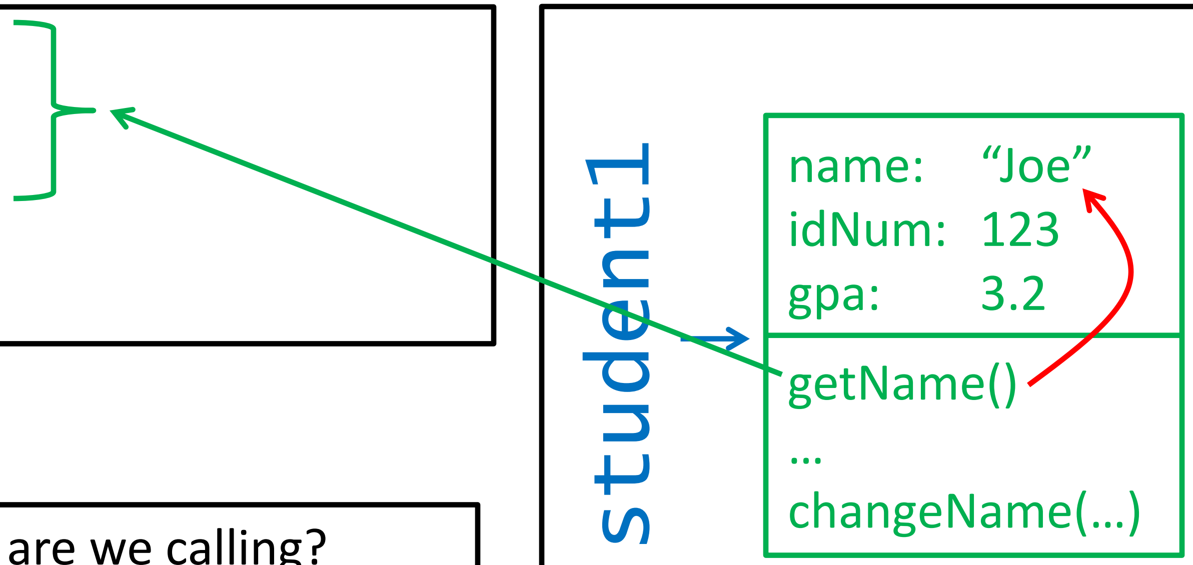
What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

So go to the object that `student1` is pointing to and look at the `getName()` method.

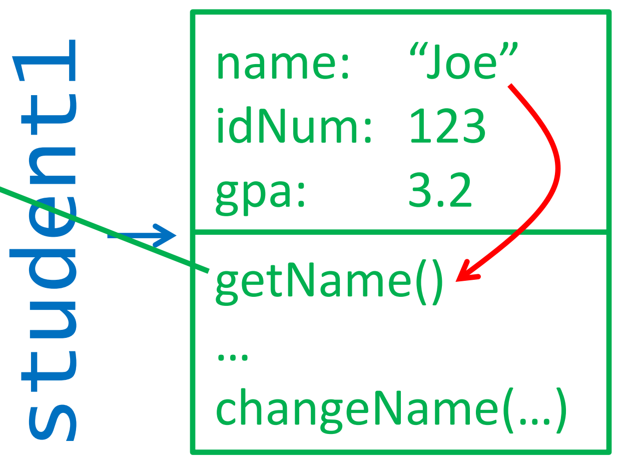
Computer Memory



Student Java Code

```
public String getName()
{              “Joe”
    return ~~name~~;
}
```

Computer Memory



What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

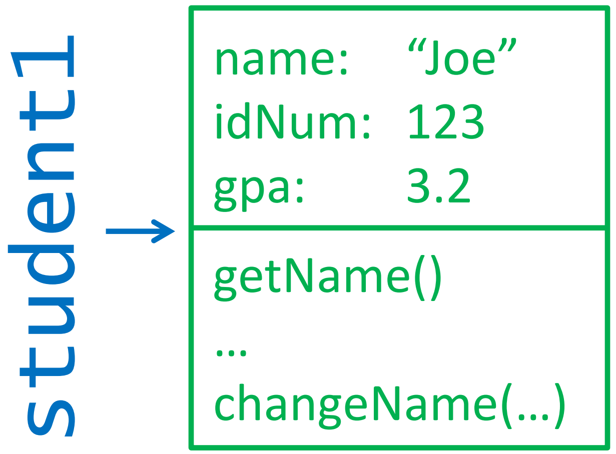
So go to the object that `student1` is pointing to and look at the `getName()` method.

The value “Joe” is returned to whoever called the method (the Driver).

Driver Java Code

```
Student student1;
student1 = new Student(“Joe”, 123, 3.2);
System.out.println(~~student1.getName()~~);
                              “Joe”
```

Computer Memory



What Happens?

Whose `getName()` method are we calling?

The object that `student1` is pointing to.

So go to the object that `student1` is pointing to and look at the `getName()` method.

The value “Joe” is returned to whoever called the method (the Driver).

Joe is printed.

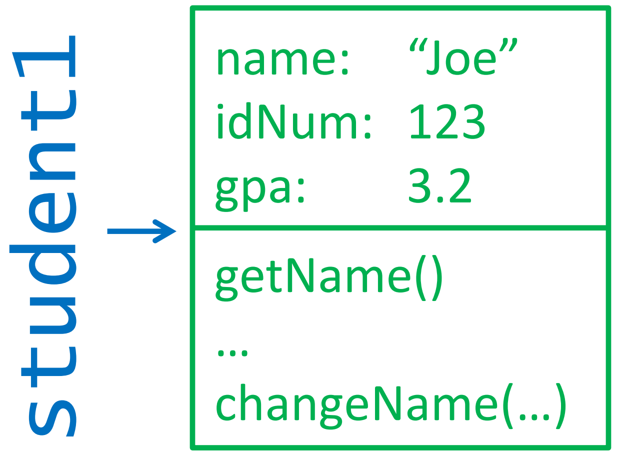
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
```

What Happens?

Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

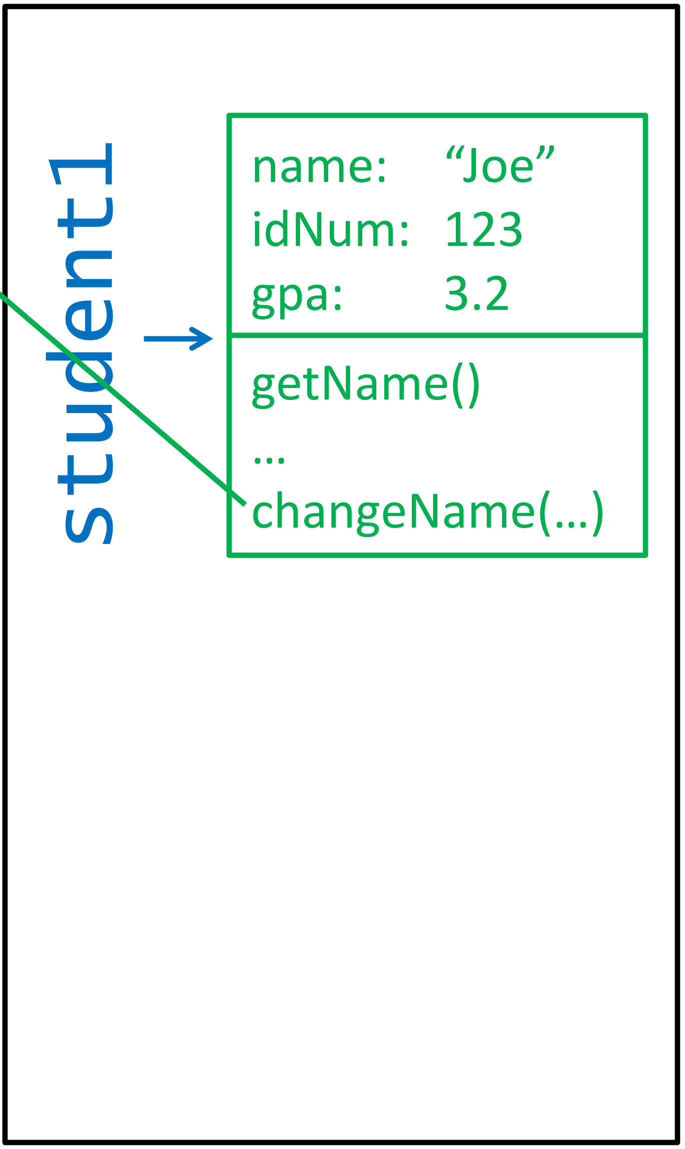
Computer Memory



Student Java Code

```
public void changeName(String newName)
{
     name = newName;
}
```

Computer Memory



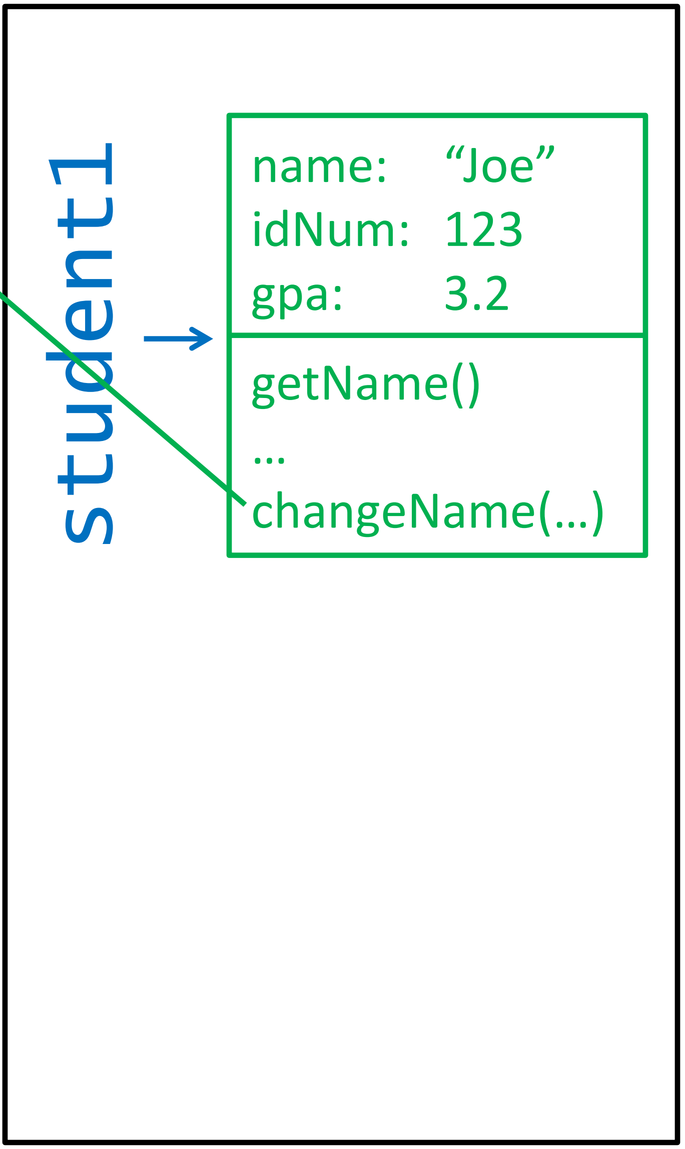
What Happens?

Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

Student Java Code

```
public void changeName(~~String newName~~)
{                          "Joseph"
     name = newName;
}
```

Computer Memory



What Happens?

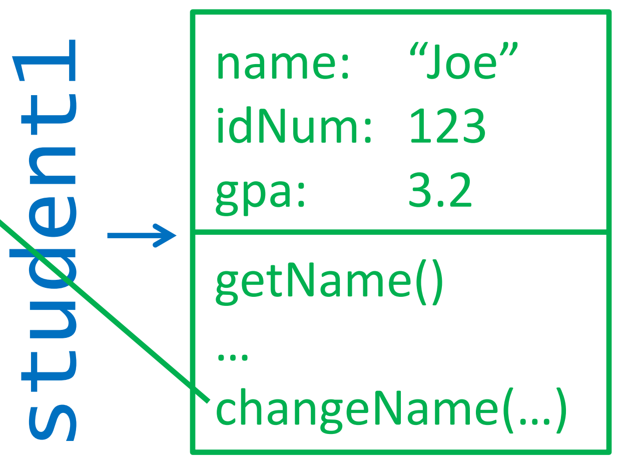
Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

`newName` contains the value "`Joseph`".

Student Java Code

```
public void changeName(~~String newName~~)
{                                   "Joseph"
    name = ~~newName~~;
}          "Joseph"
```

Computer Memory



What Happens?

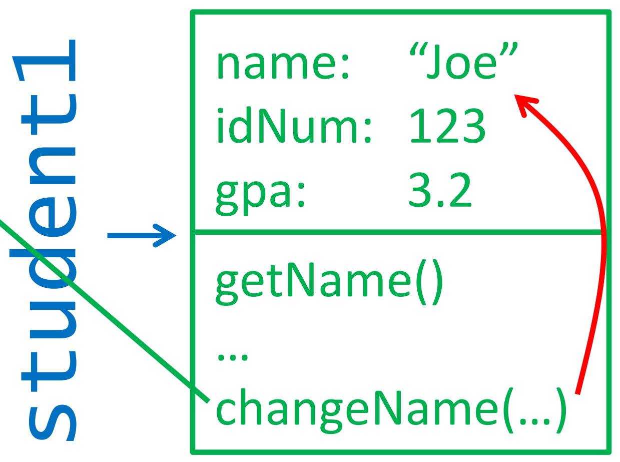
Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

`newName` contains the value "`Joseph`".

Student Java Code

```
public void changeName(~~String newName~~)
{                               “Joseph”
    name = ~~newName~~;
}          “Joseph”
```

Computer Memory



What Happens?

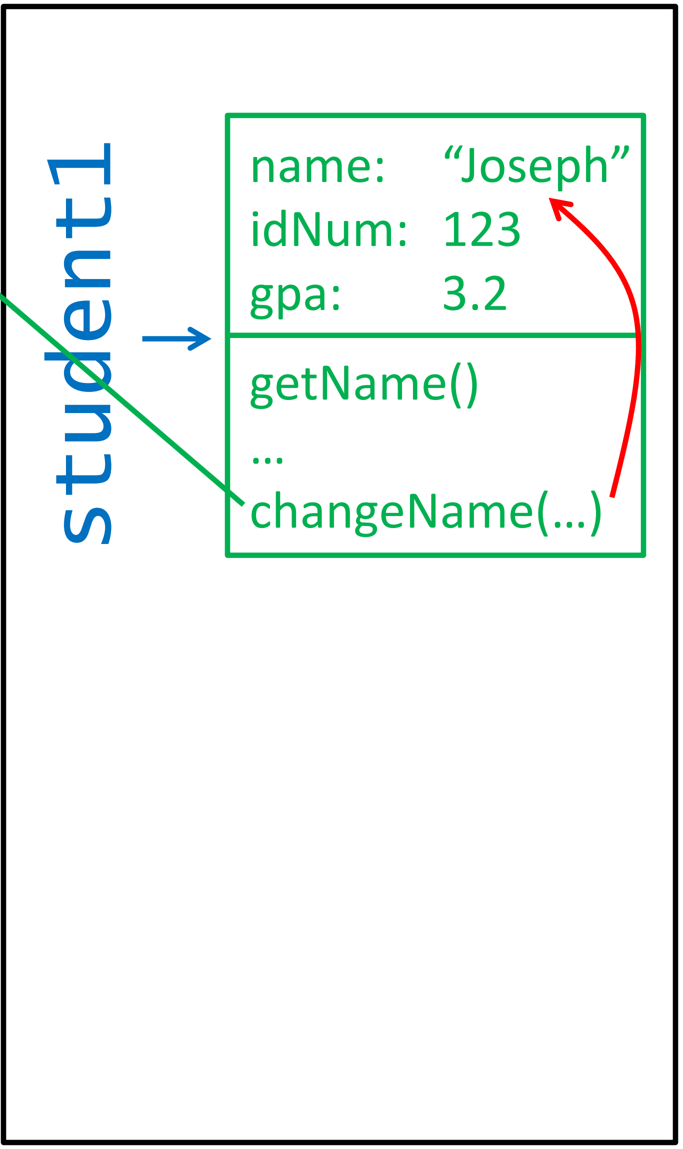
Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

`newName` contains the value “`Joseph`”.

Student Java Code

public void changeName(~~String newName~~)
{   “Joseph”
    name = ~~newName~~;
}   “Joseph”

Computer Memory



What Happens?

Whose `changeName(parameter)` method are we calling? The object student1 is pointing to. So go to the object student1 is pointing to and look at the `changeName(parameter)` method.

`newName` contains the value “`Joseph`”.

The variable name, in the object `student1` is pointing to, is changed to “`Joseph`”.

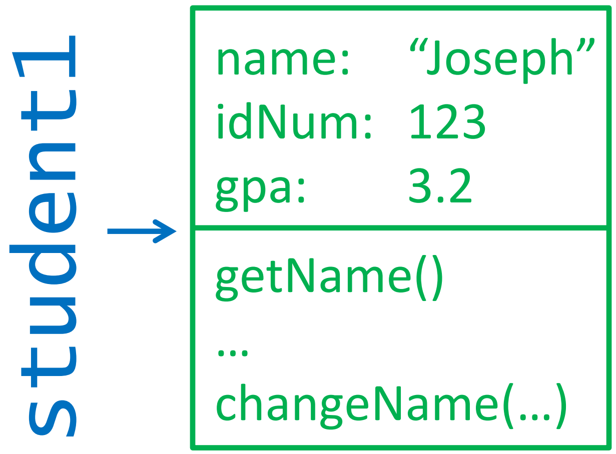
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
System.out.println(student1.getName());
```

What Happens?

The `getName()` method is called on the object that `student1` is pointing to. Thus, the current value that is in the name variable is returned to whoever asked (the Driver). The Driver then prints this value which is "`Joseph`".

Computer Memory

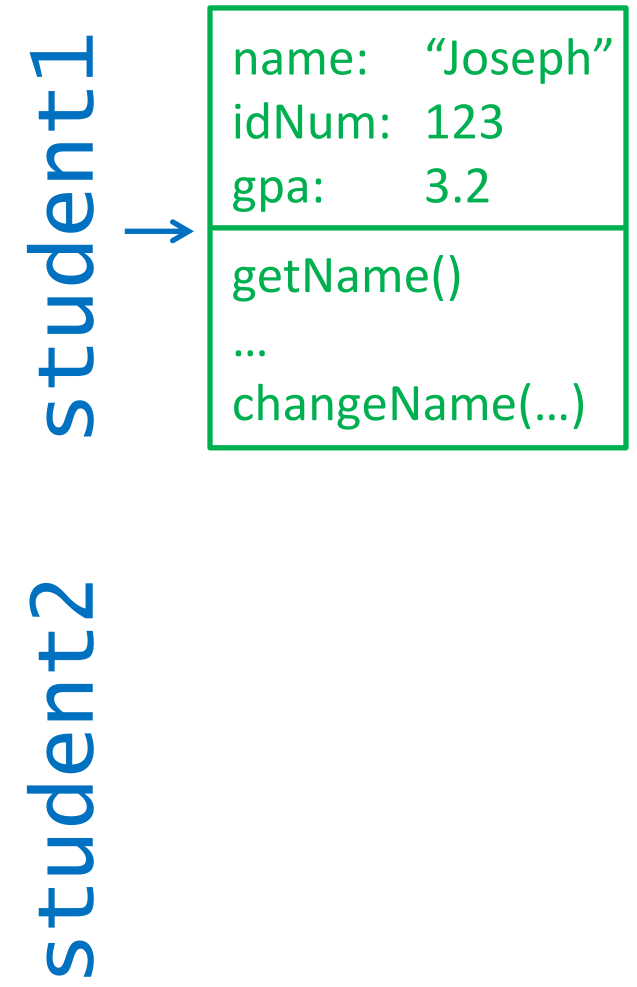


Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
System.out.println(student1.getName());

Student student2 = new Student("Sally", 321, 3.7);
```

Computer Memory



What Happens?

1. Variable Declaration.

Driver Java Code

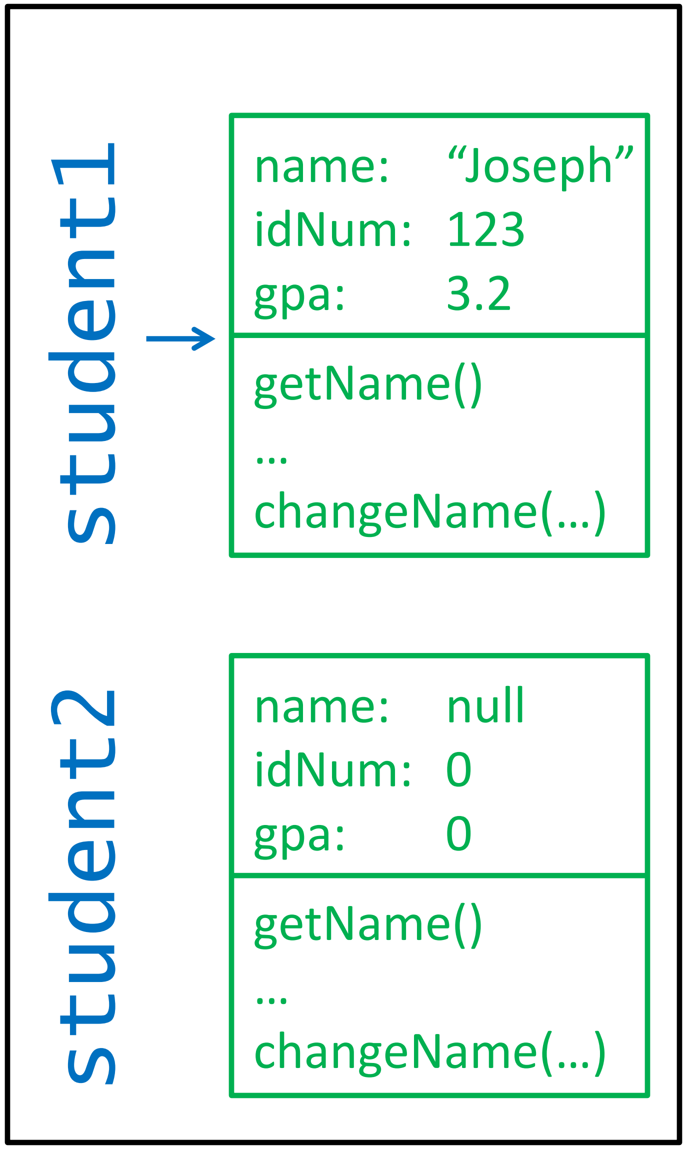
```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
System.out.println(student1.getName());

Student student2 = new Student("Sally", 321, 3.7);
```

What Happens?

1. Variable Declaration.
2. Object Instantiation.

Computer Memory



Driver Java Code

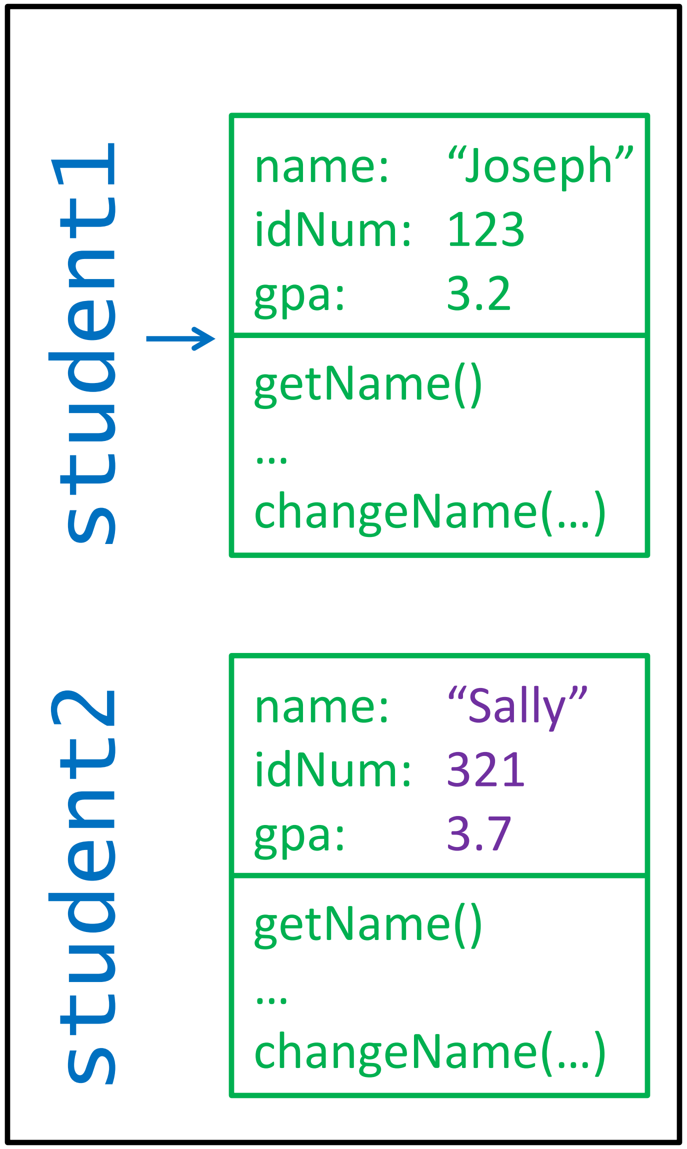
```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
System.out.println(student1.getName());

Student student2 = new Student("Sally", 321, 3.7);
```

What Happens?

1. Variable Declaration.
2. Object Instantiation.
3. Object Initialization.

Computer Memory



Driver Java Code

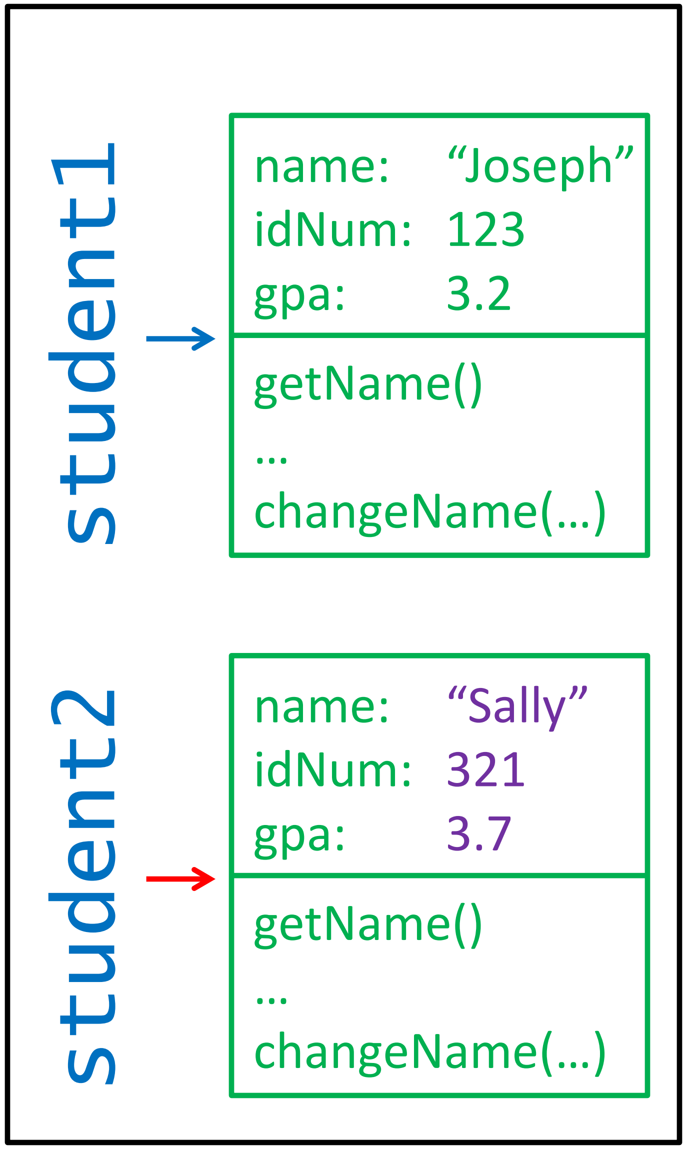
```
Student student1;
student1 = new Student(“Joe”, 123, 3.2);
System.out.println(student1.getName());
student1.changeName(“Joseph”);
System.out.println(student1.getName());

Student student2 = new Student(“Sally”, 321, 3.7);
```

What Happens?

1. Variable Declaration.
2. Object Instantiation.
3. Object Initialization.
4. Variable Assignment.

Computer Memory



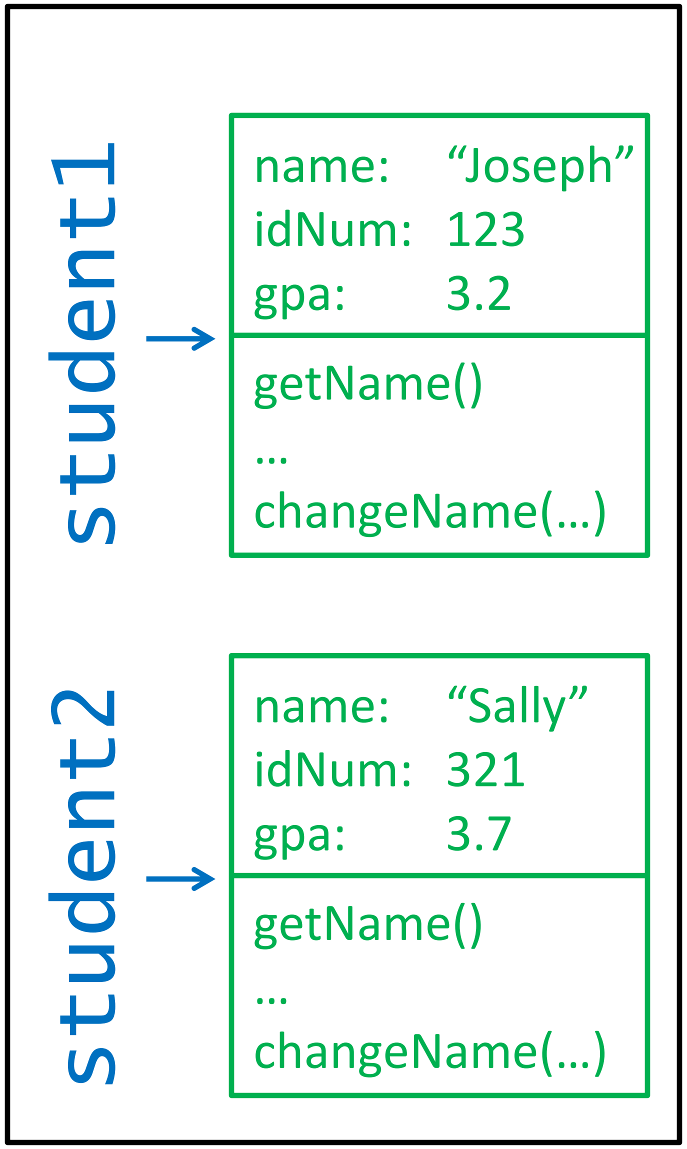
Driver Java Code

```
Student student1;
student1 = new Student("Joe", 123, 3.2);
System.out.println(student1.getName());
student1.changeName("Joseph");
System.out.println(student1.getName());

Student student2 = new Student("Sally", 321, 3.7);
```

What Happens?

Computer Memory



Exercises

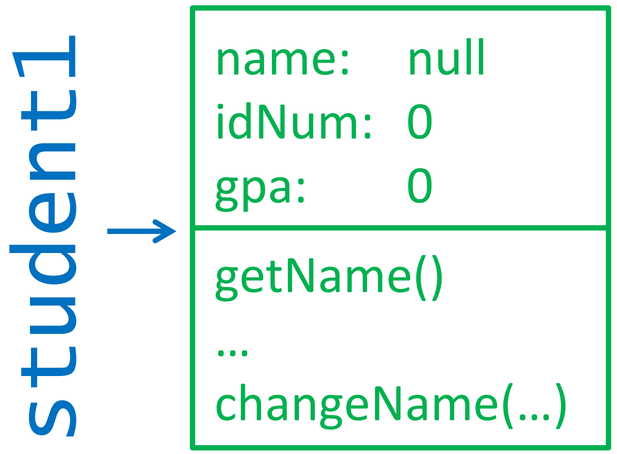
Driver Java Code

```
Student student1 = new Student(“Joe”, 123, 3.2);
```

Student Java Code

```
public Student(String inName, int inID, double iG)
{
    name = inName;
    idNum = idNum;
    gpa = gpa;
}
```

Computer Memory



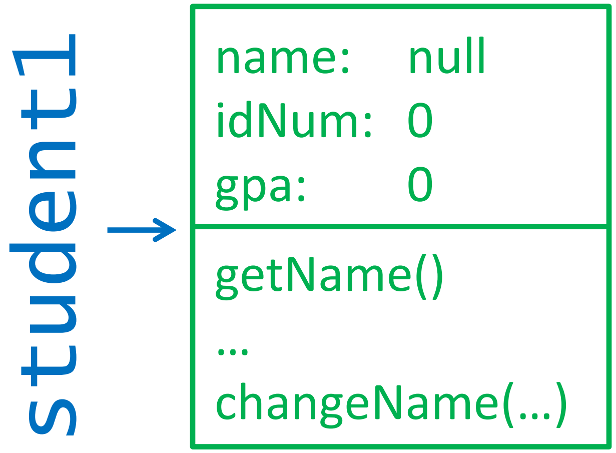
Driver Java Code

```
Student student1 = new Student(“Joe”, 123, 3.2);
```

Student Java Code

```
public Student(String inName, int inID, double iG)
{
    inName = name;
    idNum = inID;
    gpa = iG;
}
```

Computer Memory



Driver Java Code

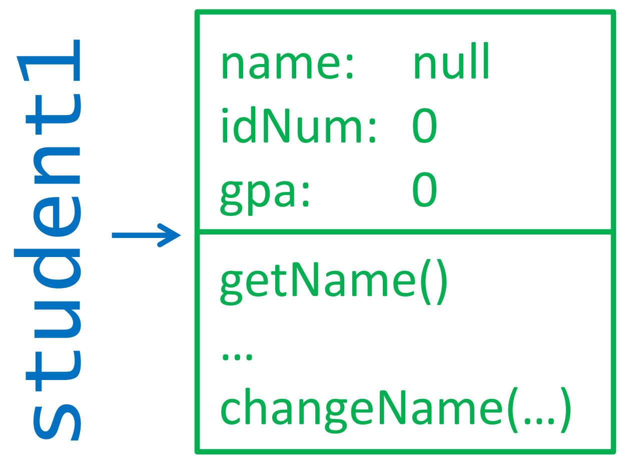
```
Student student1 = new Student("Joe", 123, 3.2);
```

Student Java Code

```
public Student(String name, int idNum, double gpa)
{
    name = name;
    idNum = idNum;
    gpa = gpa;
}
```

A little trickier. Remember that when Java sees the variable "name" inside the constructor, it will consider the variable "name" defined there, if there is one and not the instance variable "name".

Computer Memory


student1
name: null
idNum: 0
gpa: 0
getName()
…
changeName(…)