Object Instantiation and Initialization in Java

CSCI 111
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.
Student student1;
student1 = new Student("Joe", 123, 3.2);
Driver Java Code

```java
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**.
Driver Java Code

```java
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

```
name: null
idNum: 0
gpa: 0

getName()
...
changeName(...)
```
Student Java Code

```java
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

```
name: null
idNum: 0
gpa: 0
getName()
...
changeName(...)
```
**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**.
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**.
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**.
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**.

```java
Student student1;
student1 = new Student("Joe", 123, 3.2);
```
Driver Java Code

```java
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**.
3. `student1` is set to point to this new object. This step is called **Variable Assignment**.
Driver Java Code

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

Computer Memory

```
name: “Joe”
idNum: 123
gpa: 3.2
getName()
...
changeName(...)  
```

What Happens?

Whose `getName()` method are we calling?
Driver Java Code

```java
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.

Computer Memory

```
name:  "Joe"
idNum:  123
gpa:   3.2

getName()
...
changeName(...)```
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.
Student Java Code

```java
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.
Student Java Code

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

Computer Memory

```
name: “Joe”
idNum: 123
gpa: 3.2
```

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

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

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

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

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

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”.
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”.

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

```
name: “Joe”
idNum: 123
gpa: 3.2
getNamme() 
... 
changeName(...) 
```
Student Java Code

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

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”.

Computer Memory

student1

name: “Joe”
idNum: 123
gpa: 3.2
getName()
... changeName(...)

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”.

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

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

```java
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.
Driver Java Code

```java
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

```
name: "Joseph"
idNum: 123
gpa: 3.2
getName()
... changeName(...)
```

```
name: null
idNum: 0
gpa: 0
getName()
... changeName(...)
```
Driver Java Code

```java
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

```
<table>
<thead>
<tr>
<th>student1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name:</td>
<td>&quot;Joseph&quot;</td>
</tr>
<tr>
<td>idNum:</td>
<td>123</td>
</tr>
<tr>
<td>gpa:</td>
<td>3.2</td>
</tr>
<tr>
<td>getName()</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>changeName(...)</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>student2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name:</td>
<td>&quot;Sally&quot;</td>
</tr>
<tr>
<td>idNum:</td>
<td>321</td>
</tr>
<tr>
<td>gpa:</td>
<td>3.7</td>
</tr>
<tr>
<td>getName()</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>changeName(...)</td>
<td></td>
</tr>
</tbody>
</table>
```
Driver Java Code

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

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

What Happens?

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

Computer Memory

```
student1
name: "Joseph"
idNum: 123
gpa: 3.2
getName()
...
changeName(...)

student2
name: "Sally"
idNum: 321
gpa: 3.7
getName()
...
changeName(...)
```
Driver Java Code

```java
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?

```
name: "Joseph"
idNum: 123
gpa: 3.2

getName()
...
changeName(...)
```

```
name: "Sally"
idNum: 321
gpa: 3.7

getName()
...
changeName(...)
```
Exercises
Driver Java Code

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

Student Java Code

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

```
name: null
idNum: 0
gpa: 0

getName()
...
changeName(...)
```
Driver Java Code

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

Student Java Code

```java
public class Student {
    private String name;
    private int idNum;
    private double gpa;

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

Computer Memory

```
name: null
idNum: 0
gpa: 0

getName()
...
changeName(...)```
Driver Java Code

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

Student Java Code

```java
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

```
name: null
idNum: 0
gpa: 0
getName()
...
changeName(...)
```