CSE 1322 Module 2 - Part 2

Classes & Objects



Object-Oriented Programming

• Object-Oriented Programming (OOP) is based on the concept of classes, from which objects are created.

- Remember:
 - If a class is a **template**.
 - Then an object is an **instance** of that template.



Classes

- When we create a class, we are creating a new complex data type.
- Classes can represent the **concept** of something in the real world, such as dogs, cars, users, etc.



Classes

- Classes are complex data types since they can hold multiple data which we call **Attributes**.
- Classes can also contain functions which we call **methods**, **behaviors** or **member functions**.



Classes – Example

```
class Student{
}
```

- Here, we have a class for **Students** which is also a new data type.
- At this point we do not have actual students in our program, but just the template or definition of a student.



Objects

- Objects are the "materialized" version of a class.
- An object use the definitions set by the class to create a single instance of it.
- Even though there may be multiple instances of the same class, each object will have its own unique set of attributes, identity, state, and behaviors unless we specify it otherwise.



Student Example

- The definition of a class will always depend on the scope of our program.
- We can be high detailed or low detailed, but we always match the scope.



Student Example

- Let's imagine that for the overall program we need to define a student class.
- The application will need to know the name of the student, the email, the grade for each assignment (10 in total), and class grade (average of those 10 assignments)
- We also need to be able to update the grade by re-calculating the average of each assignment grade.
- We also need to be able to send an email to the student.
- And lastly, we need to be able to update the grade for an assignment



Student Example

- Student Attributes
 - First Name
 - Last Name
 - Email
 - Assignment Grades (10 in total)
 - Class grade
- Behaviors
 - Update Class Grade (calculates the average of the assignments)
 - Send Email
 - Update Assignment Grade



Student Example - Skeleton

```
class Student{
    // Attributes: first name, last name, etc.

// Behaviors: update grade, send email, etc.
}
```



Student Example - Attributes

```
class Student{
    // Attributes: first name, last name, etc.
    public String first_name;
    public String last_name;
    public String email;
    public double[] Assignments;
    public double grade;

// Behaviors: update grade, send email, etc.
}
```



```
class Student{
    // Attributes: first name, last name, etc.
    public String first_name;
    public String last name;
    public String email;
    public double[] Assignments;
    public double grade;

    // Behaviors: update grade, send email, etc.
    public double updateGrade(){
    }

    public boolean sendEmail(String message){
    }

    public void updateAssignment(){
    }
}
```



- We are not done yet. Part of the behavior we can include a Constructor.
- With the constructor we can specify what happens whenever we create an **object** or instance of this class.
- We can use the constructor to initialize values.
- The constructor works like a function since we can pass parameters.



• The syntax of a constructor looks like this:

• Like a function, the parameters are optional.



```
public Student(String first_name, String last_name, String email){
    // Initialize some attributes
    this.first_name = first_name;
    this.last_name = last_name;
    this.email = email;

    // Initialize the Assignments array
    this.Assignments = new double[10];
    // Update the grade
    this.grade = updateGrade();
}
```



- As you may have noticed, we have a new keyword: this.
- We use this keyword to resolve ambiguity of variables

```
public Student(String first_name, String last_name, String email){
    // Initialize some attributes
    first_name = first_name;
}
```

• What is first_name? a Class Attribute or a Constructor Parameter?



• Whatever expression we designate as this. will always refer to the class expression.

```
public Student(String first_name, String last_name, String email){
    // Initialize some attributes
    this.first_name = first_name;
}
```

- So, this.first_name is the Class attribute and first_name is the constructor parameter.
- Notice also that we have a dot operator "." after this. KENNESAW STA

- Let's also discuss why some class attribute are assigned with the constructor's parameter and some do not.
- Ultimately, the decision of which attributes should be initialized by parameter and which ones not comes down to the purpose and design of the class and constructor.
- But we can follow certain guidelines



- If we know that an attribute of an object will vary for every initialized object, then we can initialize this attribute by parameter value.
- For example, in the case of the **Student** class:
 - Not every student will have the same first and last name.
 - Not every student will have the same email.



- But ultimately, it will come down to design and some assumptions.
- For our use case, we add a student at the beginning of the course, so *Assignments* should all start as 0, since we do not know yet their grade.
- Also, following the previous assumption grade should be initialized as 0.0.
- In the Student class though, we are using the *updateGrade()* function. There is no other reason to do this other than to show you that we can also call functions inside the constructor.



```
public Student(String first_name, String last_name, String email){
    // Initialize some attributes
    this.first_name = first_name;
    this.last_name = last_name;
    this.email = email;

    // Initialize the Assignments array
    this.Assignments = new double[10];
    // Update the grade
    this.grade = updateGrade();
}
```



```
class Student{
    // Stabutes: first name, last name, etc.
    public String inst_name;
    public String inst_name;
    public String inst_name;
    public String manl;
    public doubled | Assignments;
    public doubled | Assignments;
    public doubled | Assignments;
    public Student(String first_name, String last_name, String email){
        // Institution extributes
        this.first_name = first_name;
        this.first_name = first_name;
        this.string_name = last_name;
        this.string_name = name double[10];
        // Undried the draw and conduction of this.string_name = name double[10];
        // Undried the draw and conduction of this.string_name = name double[10];
        // Behaviors: update grade, sand email, etc.
        public void updateGrade();
    }
    public void updateAssignment(){
    }
}

public void updateAssignment(){
}
```



• Let's now finish implementing the functions.

```
// Behaviors: update grade, send email, etc.
public double updateGrade(){
}
public boolean sendEmail(String message){
}
public void updateAssignment(){
}
```



updateGrade()

```
public double updateGrade(){
    double sum = 0;

    for(double assignment : Assignments){
        sum += assignment;
    }

    grade = sum / Assignments.length;
    return grade;
}
```



sendEmail()

```
public boolean sendEmail(String message){
    // We do not have a real email system,
    // so we will just print the message
    System.out.println("Sending an email to " + email);
    System.out.println("Message:\n" + message);
    return true;
}
```



updateAssignment()

```
public void updateAssignment(int index, double grade){
    Assignments[index] = grade;
    updateGrade();
}
```



- Our **Student** class is now ready to be used!
- To initialize an object of Student, we first need a Driver class.
- The Driver class is just the Main class.
- As previously explained:
 - The **Driver** will always contain the **Main** method (starting point of the program).
 - The name of the Driver class needs to always match with the name of the file.
 - The name of the **Driver** should always start with the **first letter capitalized**.



- The driver for this example will is going to be in our github page under the Week-4 directory.
- The name of the file will be **Main**.java.
- We can define classes within the same file, just make sure only the Driver class should have the access modifier of public.
- But we do recommend to have the **Driver** class and any other classes in separate files.



• Driver

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



 Initializing an object is very similar to initializing any other primitive type.

```
<data type> <identifier>;
<data type> <identifier> = new <data type>(<parameters>);
```

• As before, parameters are optional.



• Driver

```
public class Main {
    public static void main(String[] args) {
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
    }
}
```



- Notice that we have a new keyword: new
- This keyword is used to whenever we want to create or initialize a new instance or new object, hence *new*.
- Whenever we use this keyword, we are calling the constructor of the class.
- It also orders the computer to allocate the memory space required to save the object in your RAM.



• Driver

```
public class Main {
    public static void main(String[] args) {
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
    }
}
```



Driver

public class Main {

public static void main(String[] args) {
 Student s1;
 Student s2 = new Student("John", "Doe", "jdoe")
}

• This is just an over-simplification of how it works in memory

Memory (RAM)

	address	value);
ı	0 (s1 reference)	null	
6	1 (s2 reference)	->2	
	2 (s2 object)	s2	
	3		
	4		



```
public class Main {
    public static void main(String[] args) {
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
    }
}
```

address	value
0	
1	
2	
3	
4	



```
public class Main {
    public static void main(String[] args) {
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
    }
}
```

address	value
0	
1	
2	
3	
4	



address	value
0	
1	
2	
3	
4	



```
aublic class Main {
    public static void main (String | args)
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
    }
}

address value

0 (s1 reference) null
```

address value

0 (s1 reference) null

1

2

3

4













```
| Cartific Stories (Inst. stories, 10st stor
```



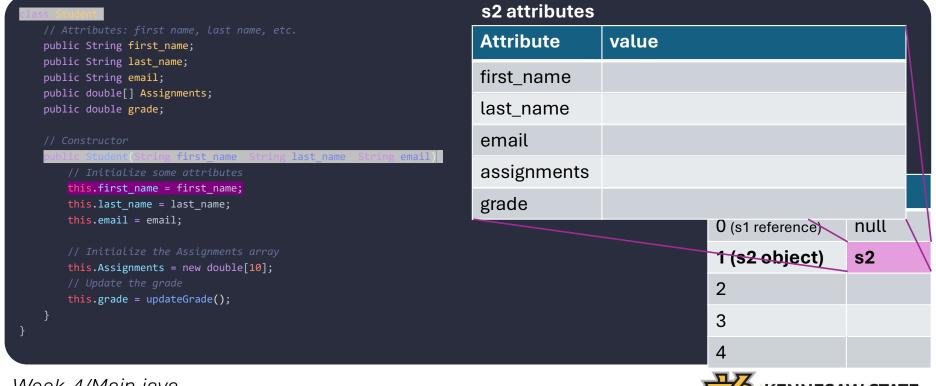
```
// Attributes: first name, last name, etc.
public String first_name;
public String last_name;
public String email;
public double[] Assignments;
public double grade;

// Constructor
public Student(String first_name, String last_name, String email){
    // Initialize some attributes
    this.first_name = first_name;
    this.last_name = last_name;
    this.email = email;

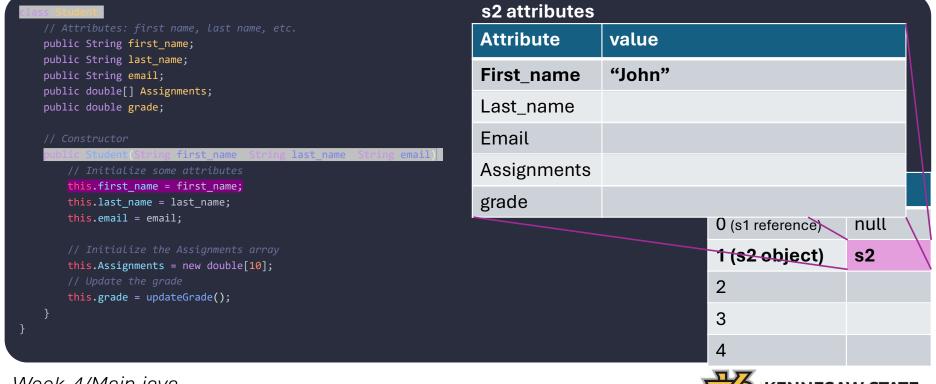
// Initialize the Assignments array
    this.Assignments = new double[10];
    // Update the grade
    this.grade = updateGrade();
}
```

address	value
0 (s1 reference)	null
1 (s2 object)	
2	
3	
4	

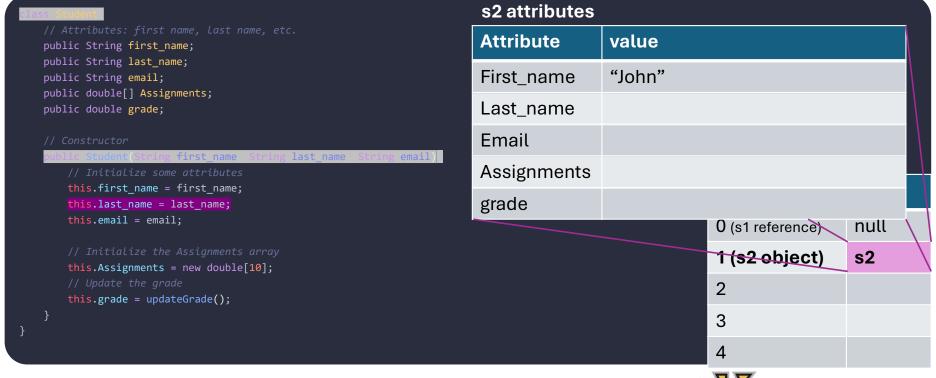












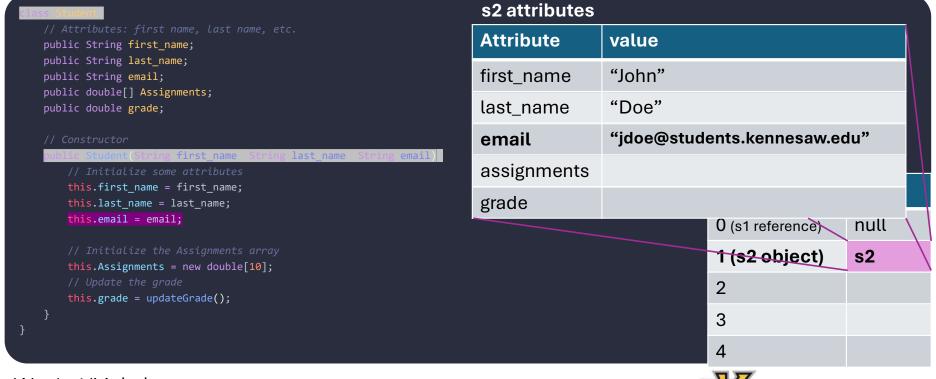




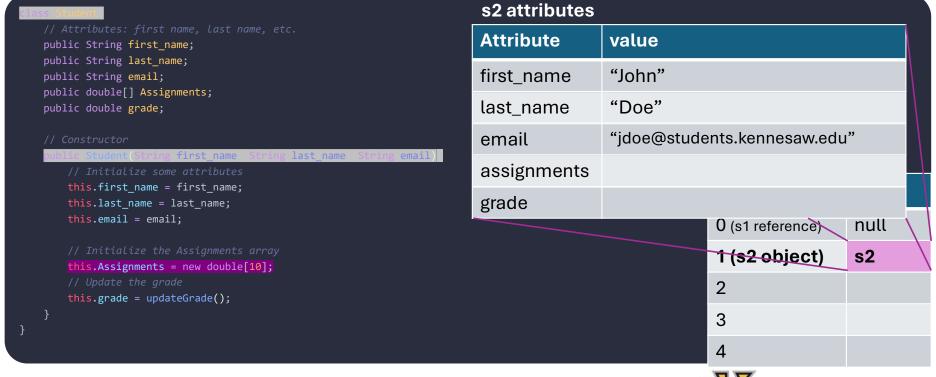




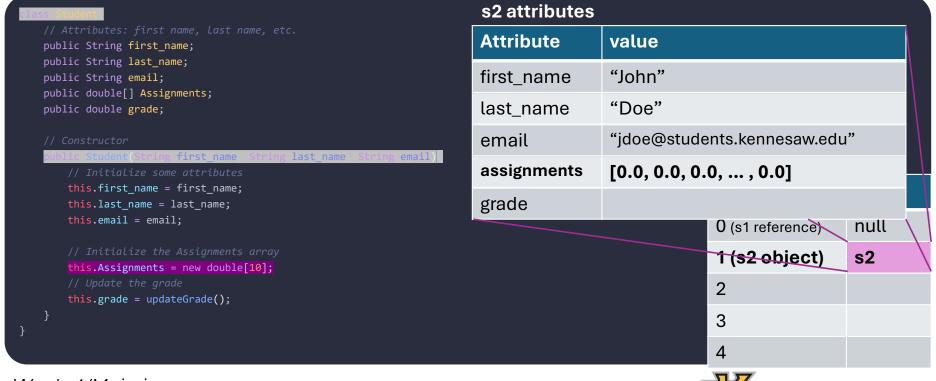




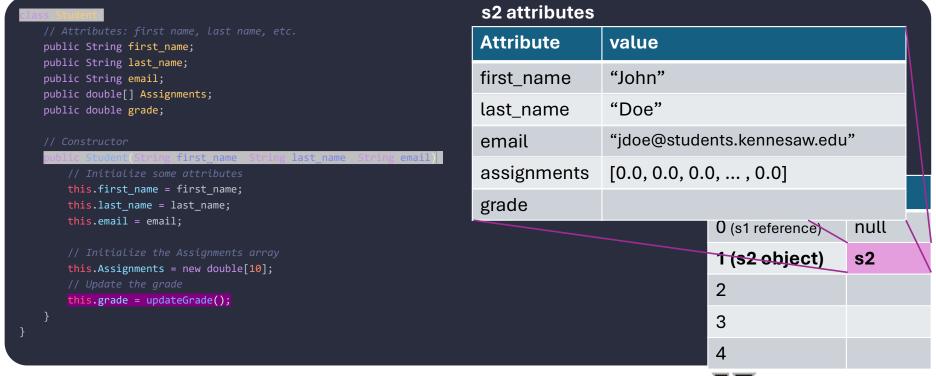




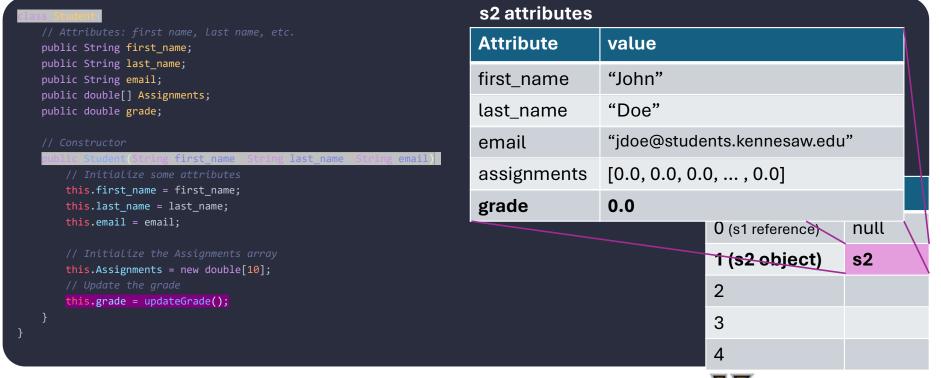




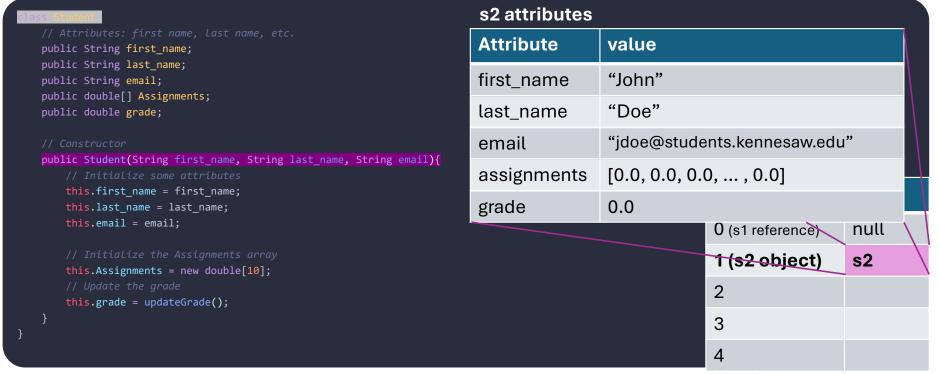








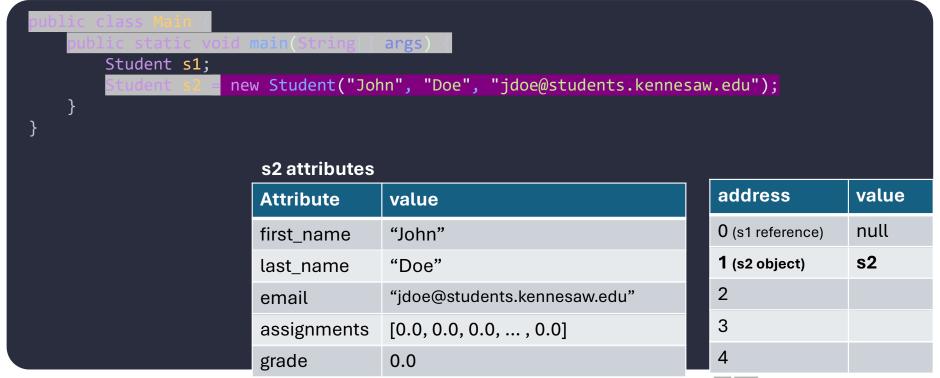






```
s2 attributes
class Student{
                                                                   Attribute
                                                                                      value
   public String first_name;
   public String last_name;
                                                                                      "John"
                                                                   first name
   public String email;
   public double[] Assignments;
                                                                                      "Doe"
                                                                   last_name
   public double grade;
                                                                                      "jdoe@students.kennesaw.edu"
                                                                   email
   public Student(String first_name, String last_name, String email){
                                                                   assignments
                                                                                      [0.0, 0.0, 0.0, \dots, 0.0]
      this.first_name = first_name;
                                                                                      0.0
                                                                   grade
      this.last_name = last_name;
      this.email = email;
                                                                                                      0 (s1 reference)
                                                                                                                          null
                                                                                                      1 (s2 object)
                                                                                                                          s2
      this.Assignments = new double[10];
      this.grade = updateGrade();
```







```
bublic static void main(String[] args) {
    Student s1;
    Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
}
```

s2 attributes

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	$[0.0, 0.0, 0.0, \dots, 0.0]$
grade	0.0

address	value
0 (s1 reference)	null
1 (s2 object)	s2
2	
3	
4	



```
public static void main(String[] args) -
Student s1;
Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
}
```

s2 attributes

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0

address	value
0 (s1 reference)	null
1 (s2 object)	s2
2 (s2 reference)	->1
3	
4	



```
Student s1;
Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
}
```

s2 attributes

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	$[0.0, 0.0, 0.0, \dots, 0.0]$
grade	0.0

address	value
0 (s1 reference)	null
1 (s2 object)	s2
2 (s2 reference)	->1
3	
4	



Student Example - Dot operator

- We have just demonstrated how to define a class and how to create an object.
- Now we need to "use" the object.
- We can use the dot operator . to access and call any visible attribute or any visible function.



Student Example – Access Modifiers

```
Different class
                                        Different package
                                                              Unrelated class
                                                                               Different module
                      but same package
                                        but subclass
                                                              but same module
                                                                               and p1 not exported
package p1;
                      package p1;
                                        package p2;
                                                              package p2;
                                                                               package x;
public class A {
                      class B {
                                        class C extends A { class D {
                                                                               class E {
    private int i;
    int j;
    protected int k;
    public int l;
                      }
```

Accessible

Inaccessible



Student Example - Dot operator

On the same Student class example, lets try to use the sendEmail()
function on the s2 object.

```
public class Main {
   public static void main(String[] args) {
        Student s1;
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");

        if(s2.sendEmail("Hello, John!")){
            System.out.println("Email sent successfully!");
        }
        else{
            System.out.println("Failed to send the email!");
        }
   }
}
```



Student Example - Dot operator

sendEmail()

```
public boolean sendEmail(String message){
    // We do not have a real email system,
    // so we will just print the message
    System.out.println("Sending an email to " + email);
    System.out.println("Message:\n" + message);
    return true;
}
```



Student Example – Dot operator

```
Sending an email to jdoe@students.kennesaw.edu
Message:
Hello, John!
Email sent successfully!
```

Student Example - Dot operator

• Remember that anything set as **visible** can be accessed with the dot operator.

```
System.out.println(s2.first_name + " " + s2.last_name);
John Doe
```



Overloaded Constructors

- Like functions, we can also have overloaded constructors.
- We follow the same premise; we can have multiple constructors within a class, but they all must have a different parameter signature.



Overloaded Constructors – Student Example

- Let's change the scope of our **Student** class.
- This is something that will happen a lot within a project. Since we have implemented a class, it will be easier to modify.



Overloaded Constructors – Student Example

- The scenario is this:
 - We had an issue with the database in the middle of the semester.
 - This database stored all the student data, and everything got erased.
 - Luckily, we had a backup of this data and where able to fix it this issue by creating a new database.
 - The problem is that now we need to migrate this data into the database manually.
 - Therefore, we need to be able to initialize a new student with some assignment grades.



Overloaded Constructors – Student Example

 To accommodate this new scope, we are going to add another constructor that is going to enable us initializing a **Student** object with already existing Assignment grades.

```
public Student(String first_name, String last_name, String email, double[] Assignments){
    // Initialize some attributes
    this.first_name = first_name;
    this.last_name = last_name;
    this.email = email;
    this.Assignments = Assignments;

// Update the grade
    this.grade = updateGrade();
}
```

Overloaded Constructors

Week-4/Student1.java



Overloaded Constructors - Chaining

- Some of the code being executed in the overloaded constructor is very similar to the first constructor.
- As an "improvement", we can chain the first constructor within the new constructor.
- We can use the this keyword to call a constructor of the class.



Overloaded Constructors - Chaining

```
class Student{
    // Constructor
    public Student(String first_name, String last_name, String email){
        // Initialize some attributes
        this.first_name = first_name;
        this.last_name = last_name;
        this.email = email;

        // Initialize the Assignments array
        this.Assignments = new double[10];
        // Update the grade
        this.grade = updateGrade();
}

// Overloaded constructor
public Student(String first_name, String last_name, String email, double[] Assignments){
        // Initialize some attributes
        this(first_name, last_name, email);

        this.Assignments = Assignments;

        // Update the grade
        this.grade = updateGrade();
}
```

Week-4/Student1.java



Default Constructor

- Every class will have a "default" constructor.
- This constructor will "exist" even if we do not specifically declare it.
- The default constructor does not take any parameters and by default, it will initialize the object and set every parameter to their default value.
 - Number types will be set to **0** or **0.0**.
 - Boolean to False.
 - Objects to **NULL**.



Default Constructor

- We can modify this behavior by declaring a default constructor.
- This will Override the "default" default constructor.

```
// Default Constructor
public Student(){
    first_name = "John";
    last_name = "Doe";
    email = "";
    Assignments = new double[10];
    grade = 0;
}
```



Default Constructor

• To call the default constructor, we must use the **new** keyword and call the constructor with no parameters.

```
Student s3 = new Student();
```



Function Override - toString()

- Every class that we create **inherits** some behaviors or methods.
- We will discuss **inheritance** later, but for now lets just assume that by "default" every class in Java contain a set of functions.
- One of these is the **toString()** function.
- This function can be used to print a "String" representation of an object.
- By default, the **toString()** function will just return a string containing the name of the object's class and its hash code

Function Override - toString()

System.out.println(s2);

Student@3f99bd52



Function Override – toString()

- We can change the behavior of this function by overriding it.
- To override it we can just define a function with the **same function header**.
- In this case:

```
class Student{
   public String toString(){
       return first_name + " " + last_name + " " + email;
   }
}
```



Function Override – toString()

- We can add an additional layer of safety by adding the **@Override** annotation.
- This does not change how the code works, but it will make sure we are overriding a method and not creating a new method.
- Whenever we add the **@Override** annotation, the compiler will check that the function right under it is overriding something.
- If it cannot find the "original" method, it will throw a compiler error.
- This annotation is just a safety mechanism.



Function Override – toString()

• In this case, this is valid since **Student** has by default a method called **toString()** that returns a **String** value.

```
class Student{
    @Override
    public String toString(){
        return first_name + " " + last_name + " " + email;
    }
}
```



Function Override - toString()

• This means that whenever we print a student object, it will print that object's first and last name attribute and its email.

```
class Student{
    @Override
    public String toString(){
        return first_name + " " + last_name + " " + email;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
        System.out.println(s2);
    }
}
```

John Doe jdoe@students.kennesaw.edu



- As mentioned previously, a good coding practice in Java is to set everything to private and only changing the access modifier if it is needed.
- Most class functions and constructors should be **public**.
- The following code will still be regarding the **Student** Class, but it will be located in our github in the *Week-4/Student/* directory.



- We are going to change the access modifier for all of the attributes to private.
- This is a safer programming practice since we want to make certain parts of this class non-accessible or accessible under certain conditions by other classes.



- For example, we can make sure that whenever we want to change the name of the student, it must be a new name.
- Or we do not want the grade attribute being able to be overridden with some value, therefore this value can only be modified by the updateGrade() function.

```
public class Main {
    public static void main(String[] args) {
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
        s2.grade = 4.0;
    }
}
```



```
class Student{
    // Attributes: first name, last name, etc.
    private String first_name;
    private String last_name;
    private String email;
    private double[] Assignments;
    private double grade;
    ...
}
```



• With this change you will notice that we cannot access the class attributes with the dot operator.

```
public class Main {
    public static void main(String[] args) {
        Student s2 = new Student("John", "Doe", "jdoe@students.kennesaw.edu");
        System.out.println(s2.first_name + " " + s2.last_name);
    }
}
```



- Since there are certain attributes that we may want to retrieve its value or change, we are going to create two types of functions: Getter and Setters.
- These are nothing special, it is just a term we label functions that retrieves the value of an attribute and functions that "set" the value to an attribute.
- We usually name these functions as get<attribute name> and set<attribute name>.



• Here is an example of a getter and setter for **first_name**.

```
class Student{
    // Getters
    public String getFirst_name(){
        return first_name;
    }

    // Setters
    public void setFirst_name(String first_name){
        this.first_name = first_name;
    }
}
```



- Since the access to first_name is being restricted by marking it as private, we use getters and setters to provide a controlled access.
- This ensures encapsulation and allows for abstracting the internal details of the **Student** class.
- Through this controlled access, we can add more complexity whenever we want to retrieve or change the attribute value.



- For example, let's say we added to the scope the logic that to change a student's first name, we want to make sure:
 - It is not an empty String.
 - It is not the same name.



```
public void setFirst_name(String first_name){
    if(first_name.isEmpty()){
        System.out.println("First name cannot be empty!");
        return;
    }
    else if(this.first_name.equals(first_name)){
        System.out.println("First name is already set to " + first_name);
        return;
    }
    this.first_name = first_name;
}
```



Getters and Setters - Student Class

```
// Getters
public String getFirst_name(){
    return first_name;
}
public String getLast_name(){
    return last_name;
}
public String getEmail(){
    return email;
}
public double[] getAssignments(){
    return Assignments;
}
public double getGrade(){
    return grade;
}
```



Getters and Setters - Student Class

```
public void setFirst_name(String first_name){
    if(first_name.isEmpty()){
        System.out.println("First name cannot be empty!");
        return;
    }
    else if(this.first_name.equals(first_name)){
        System.out.println("First name is already set to " + first_name);
        return;
    }
    this.first_name = first_name;
}

this.first_name = first_name;
}

public void setLast_name(String last_name){
    if(last_name.isEmpty())(
        System.out.println("Last name cannot be empty!");
        return;
    }
    else if(this.last_name.equals(last_name))(
        System.out.println("Last name is already set to " + last_name);
        return;
    }
    this.last_name * last_name;
}
```



Getters and Setters - Student Class

```
public void updateAssignment(int index, double grade){
    Assignments[index] = grade;
    updateGrade();
}

public double updateGrade(){
    double sum = 0;

    for(double assignment : Assignments){
        sum += assignment;
    }

    grade = sum / Assignments.length;

    return grade;
}
```



- As discussed in methods, Java treats Primitive Types in parameters as Pass by Value.
- We also discussed that for Complex Types, whenever we pass an object as parameter Java does not treat them as true Pass by Reference.
- Instead, for objects as parameter, Java implements something similar.



- Remember that for objects, when we initialize them, we create two things in memory:
 - The actual object data.
 - The reference to the address of that object.



Driver

Memory (RAM)

<pre>public class Main {</pre>	address	value	
<pre>public static void main(String[] args) {</pre>	0 (s1 reference)	null	
Student s1; Student s2 = new Student("John", "Doe", "jdoe	1 (s2 reference)	->2):
}	2 (s2 object)	s2	/ 3
This is just an over simplification of	3		
This is just an over-simplification of	4		
how it works in memory			

Week-4/Main.java



• Whenever we pass an object as a parameter, we are not passing the object but instead we are creating a copy of the reference to the address of the object.



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0	
1	
2	
3	
4	



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s 1
1	
2	
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	> 0
2	
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }

    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2	
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }

    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→ 0
2	
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→ 0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→ 0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→ 0
2 (s1 reference in update_name())	> 0
3	
4	

Attribute	value
first_name	"John"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	> 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0, , 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"James"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0, , 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→ 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0, , 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	> 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Doe"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	> 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Smith"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
}

public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"James"
last_name	"Smith"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
    public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(last_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→ 0
2 (s1 reference in update_name())	> 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Smith"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(first_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→ 0
3	
4	

Attribute	value
first_name	"James"
last_name	"Smith"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0



```
public class Main {
   public static void update_name(Student s){
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the new first name: ");
        String first_name = sc.nextLine();

        System.out.print("Enter the new last name: ");
        String last_name = sc.nextLine();

        s.setFirst_name(first_name);
        s.setLast_name(first_name);
    }
    public static void main(String[] args) {
        Student s1 = new Student("John", "Doe", "jdoe@student.kennsaw.edu");

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());

        update_name(s1);

        System.out.println(s1.getFirst_name() + " " + s1.getLast_name());
    }
}
```

address	value
0 (s1 object)	s1
1 (s1 reference)	→0
2 (s1 reference in update_name())	→0
3	
4	

Attribute	value
first_name	"James"
last_name	"Smith"
email	"jdoe@students.kennesaw.edu"
assignments	[0.0, 0.0, 0.0,, 0.0]
grade	0.0

