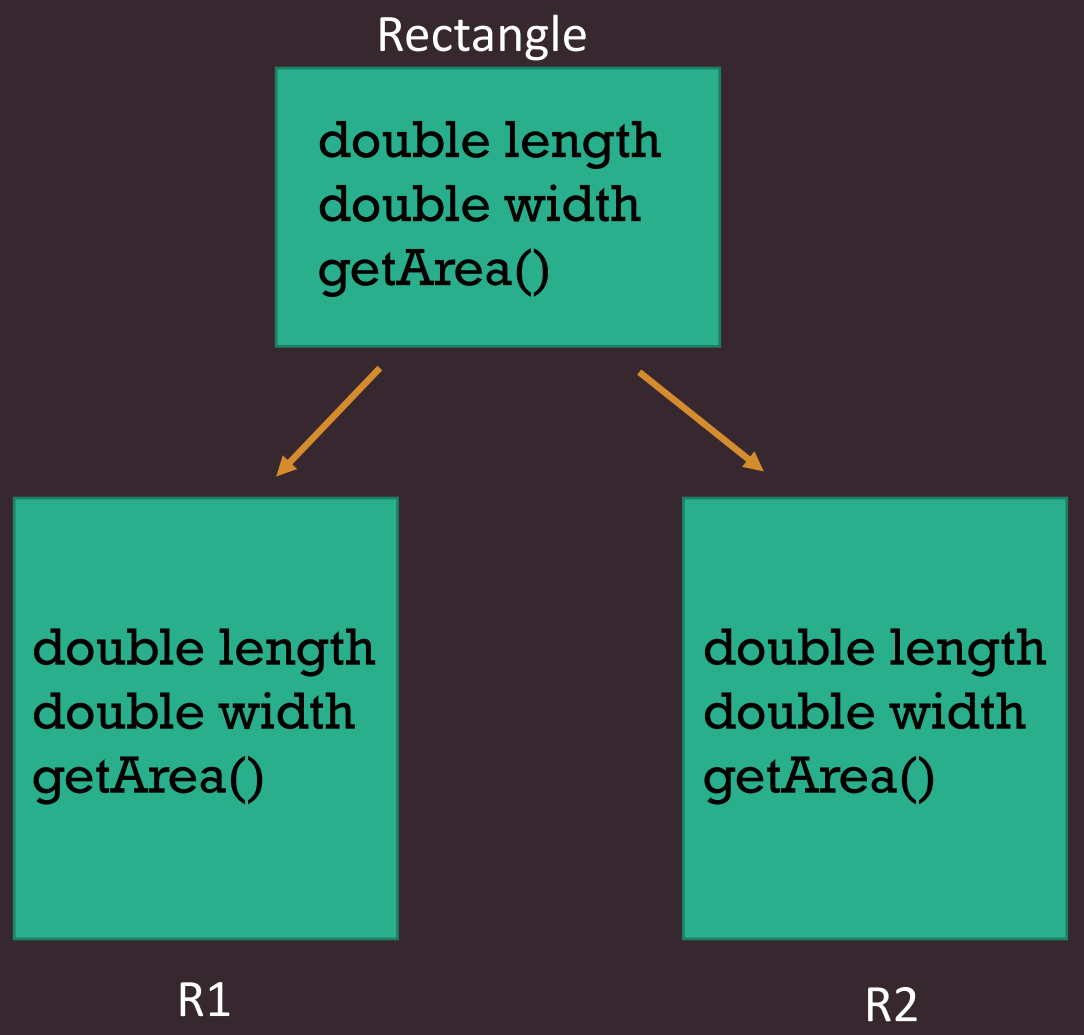
OBJECT ORIENTED PROGRAMMING (OOP)

Classes & Objects

- Class is a template/blueprint, where we define variable and methods
- Objects are the instances of the class. Each object will have its own copy of variables.



Remember this slide?

1

```
String dogName="rover";
int dogWeight=23;

if(dogWeight>20){
      System.out.println(dogName + " says WOOF WOOF");
}else{
      System.out.println(dogName + " says woof woof");
}
```

2

```
String dogName="spot";
int dogWeight=13;

if(dogWeight>20){
      System.out.println(dogName + " says WOOF WOOF");
}else{
      System.out.println(dogName + " says woof woof");
}
```

3

```
String dogName="spike";
int dogWeight=53;

if(dogWeight>20){
      System.out.println(dogName + " says WOOF WOOF");
}else{
      System.out.println(dogName + " says woof woof");
}
```

4

```
String dogName="lady";
int dogWeight=17;

if(dogWeight>20){
      System.out.println(dogName + " says WOOF WOOF");
}else{
      System.out.println(dogName + " says woof woof");
}
```

Constructors

- A constructor is a method that is automatically called when an object is created.
- Constructors are used to initialize the object's instance fields.
- They are called constructor because the help construct an object.
- new keyword invokes the constructor of the class.

Creating Constructors

- Constructor is a special method that matches the name of the class and has no return type.

```
public class Car{

    public Car(){

    }

}
```

```
public class Employee{

    public Employee(int age){

    }

}
```

Types of Constructors

- No-argument Constructor: A constructor that has no parameter is known as default constructor. If we do not define a constructor in a class, then compiler creates default constructor(with no argument) for the class. If we write a constructor with arguments or no-arguments then the compiler does not create a default constructor. Default constructor provides the default values to object like 0, null, etc. depending on the type.

- Parameterized Constructor: A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use parameterized constructor.

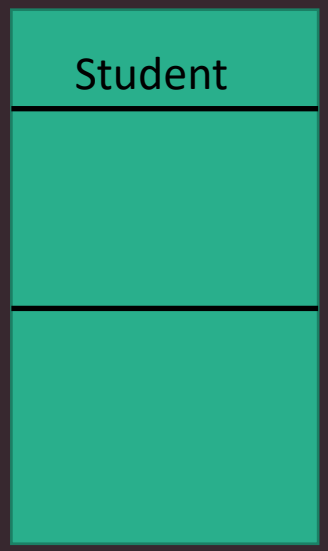
this Keyword

- this is a reference variable in Java that refers to the current object.

```
public class Test{

    int a;
    int b;

    public Test(int a,int b){

        this.a = a;
        this.b = b;

    }

}
```

Task

- For the following objects create Student class template.
- Write code for the Student class and another class to test it.
- This class has 6 properties: name,age,gender(M/F),year,course(JS,Java,TypeScript) and university



Mike 17
M 2016 Java
Cybertek University
attendLecture
submitAssignment
attendLab

student1

Smith 18
M 2015 JS
Cybertek University
attendLecture
submitAssignment
attendLab

student2

Mary 19
F 2017 TS
Cybertek University
attendLecture
submitAssignment
attendLab

student3

Constructors Overloading

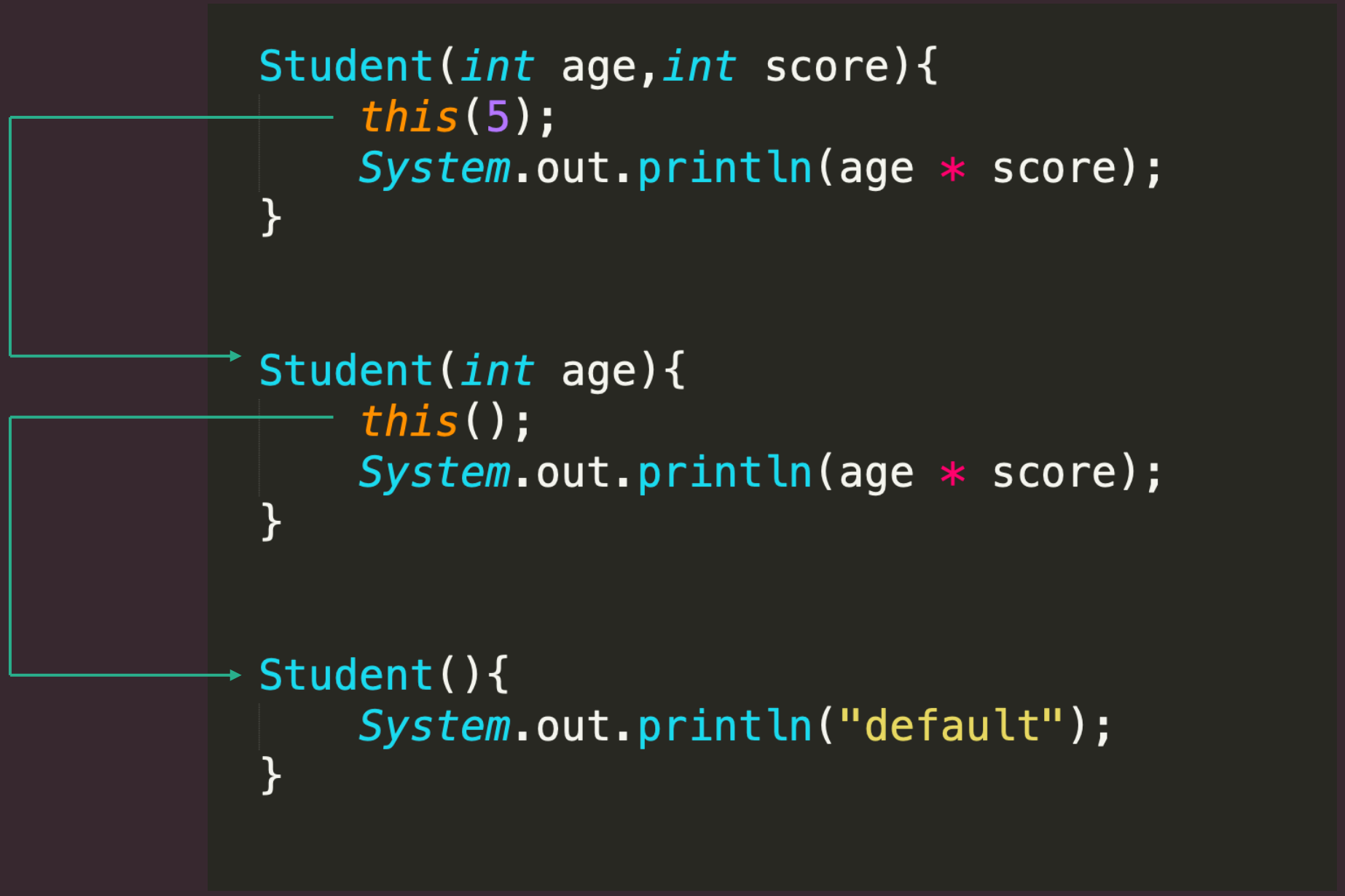
- Like methods, we can overload constructors for creating objects in different ways. Compiler differentiates constructors on the basis of numbers of parameters, types of the parameters and order of the parameters.

```java
class Student
{
    // constructor with one argument
    Student(String name)
    {
        System.out.println("Constructor with one " +
                           "argument - String : " + name);
    }

    // constructor with two arguments
    Student(String name, int age)
    {

        System.out.println("Constructor with two arguments : " +
                " String and Integer : " + name + " "+ age);

    }

    // Constructor with one argument but with different
    // type than previous..
    Student(long id)
    {
        System.out.println("Constructor with one argument : " +
                                    "Long : " + id);
    }
}
```

```java
public static void main(String[] args)
{

    // Invoke the constructor with one argument of
    // type 'String'.
    Student geek2 = new Student("Shikhar");

    // Invoke the constructor with two arguments
    Student geek3 = new Student("Dharmesh", 26);

    // Invoke the constructor with one argument of
    // type 'Long'.
    Student geek4 = new Student(325614567);
}
```

Constructors Chaining

- Constructor chaining is the process of calling one constructor from another constructor with respect to current object.
- Chaining can be done using this() keyword.
- this() keyword should always be the first line of the constructor.


Student(int age,int score){
this(5);
System.out.println(age * score);
}
Student(int age){
this();
System.out.println(age * score);
}
Student(){
System.out.println("default");
}

Difference Between Constructors and Methods

Java Constructor	Java Method
A constructor is used to initialize the variables of a Class	A method is used to define the behavior/functionalities of an object
A constructor must not have return type	The method may or may not have a return type
The constructor has invoked automatically at the time of object creation. Constructors can be called explicitly when there are multiple constructors are defined.	The method is invoked explicitly using the dot operator
The java compiler provides a default constructor if you do not have any constructor	There is no existence of default Method
Constructor name must be same as the class name	Method name may or may not be same as class name

Passing Objects as Arguments

- When an object is passed as an argument to a method, the object's address is passed into the method's parameter variable. As a result, the parameter references the object.

```
class Apple{
    String color = "red";
}


public class Main{
    public static void main(String[] args){
        Apple apple = new Apple();
        System.out.println(apple.color);

        changeApple(apple);
        System.out.println(apple.color);

    }

    public static void changeApple(Apple apple){
        apple.color = "green";
    }
}
```