DePaul University

OOP Intro and Basics

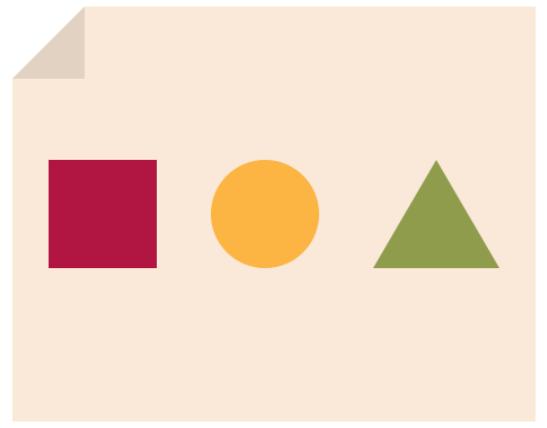
Object-oriented Software Development
SE 350– Spring 2021

Vahid Alizadeh

Week 2.1
April 6, 2021

Programming Paradigms

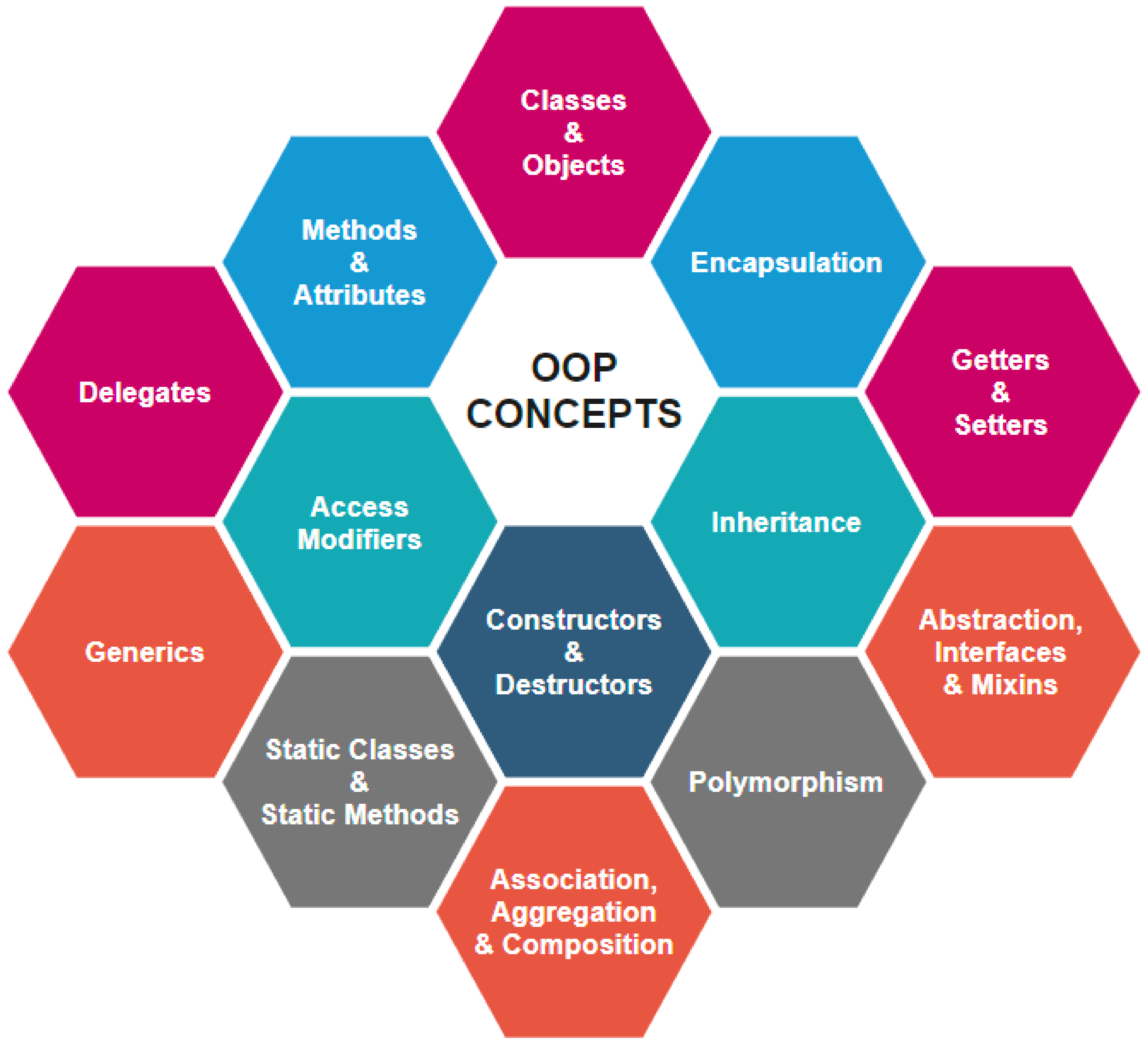
Categories of programming styles

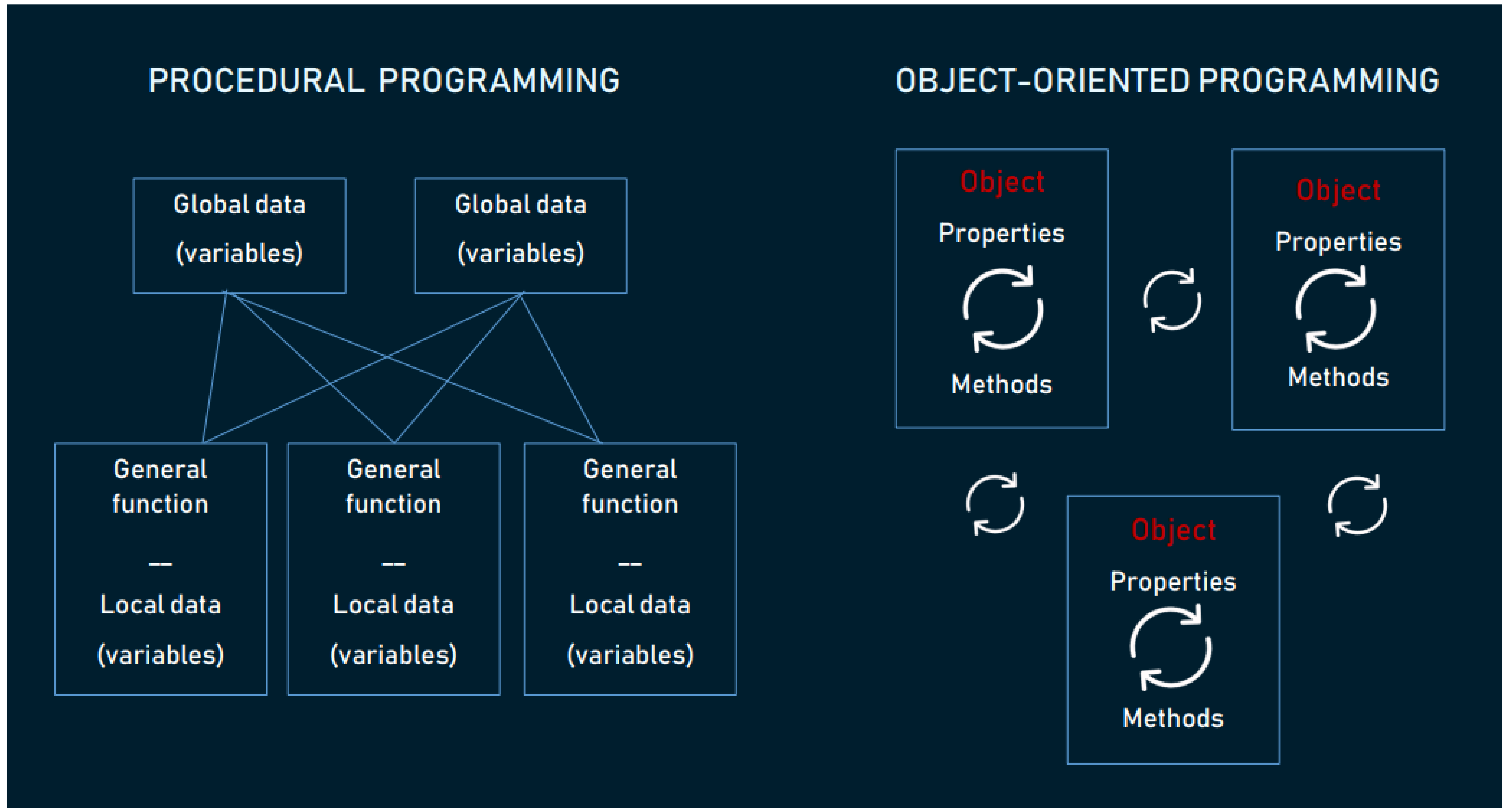


Object-oriented Programming

Introduction & Basics

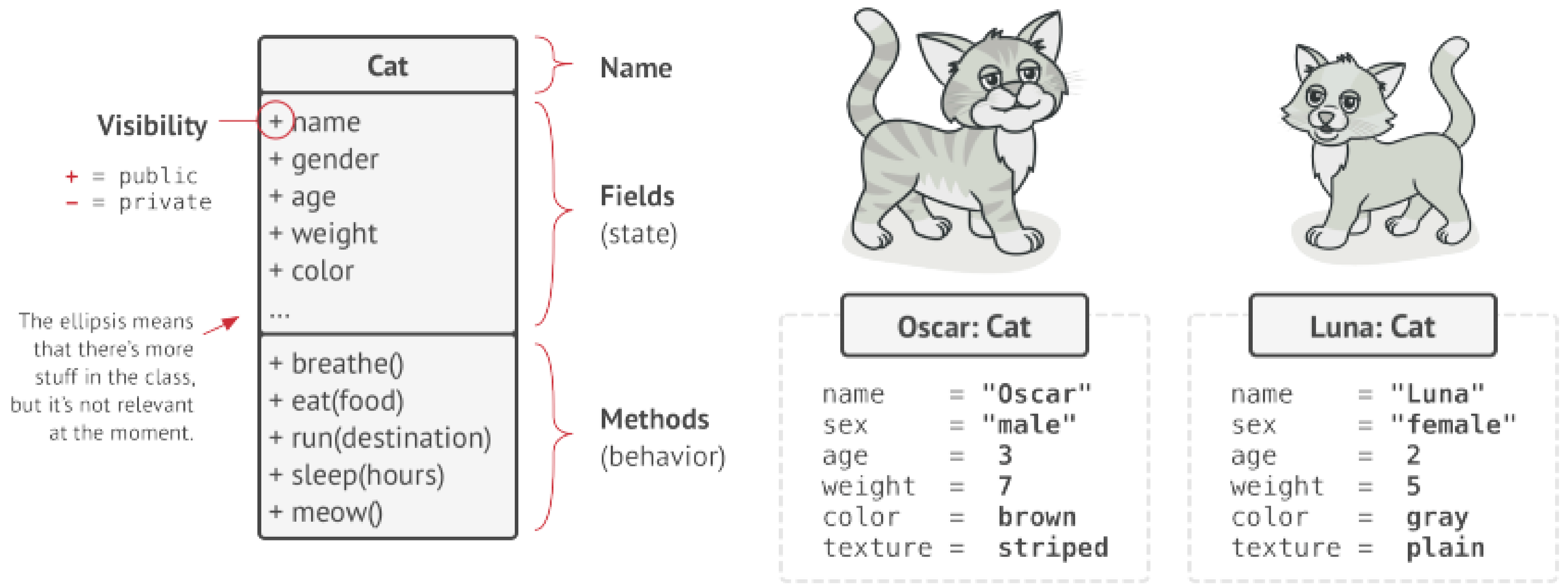
Object-oriented Programming




PROCEDURAL PROGRAMMING
Global data (variables)
Global data (variables)
General function — Local data (variables)
General function — Local data (variables)
General function — Local data (variables)
OBJECT-ORIENTED PROGRAMMING
Object
Properties
Methods
Object
Properties
Methods
Object
Properties
Methods

OOP Building Blocks: Classes and Objects

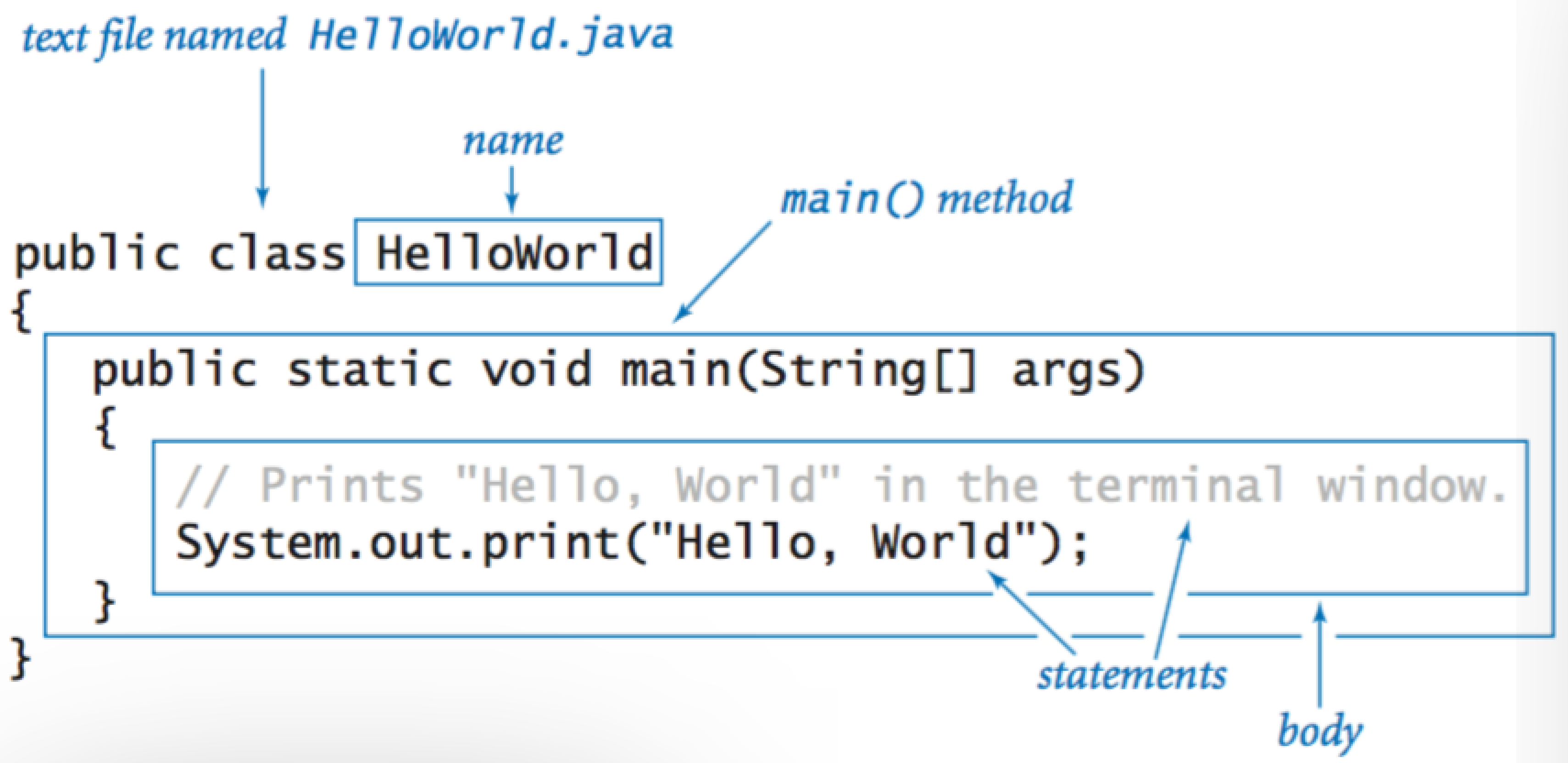
- **Class**
  - Group of objects
  - User defined data types
- **Object**
  - Object = Instance
- **3 Characteristics**
  - State, Behavior, Identity



N/A	What is it?	Information Contained	Actions	Example
Classes	Blueprint	Attributes	Behaviors defined through methods	Dog Template
Objects	Instance	State, Data	Methods	Rufus, Fluffy

OOP Building Blocks: Classes & Objects

- **Class**
  - How to create a Class?
  - structure of a class
- **Object**
  - How to create an object?
    - Declaration
      - Creating a *reference*
      - No memory allocation
    - Instantiation
    - Initialization
- **Class Members**
  - Instance variables
  - Methods



```
public class HelloWorld
{
   public static void main(String[] args)
   {
      // Prints "Hello, World" in the terminal window.
      System.out.print("Hello, World");
   }
}
```

```
type objectName;
```

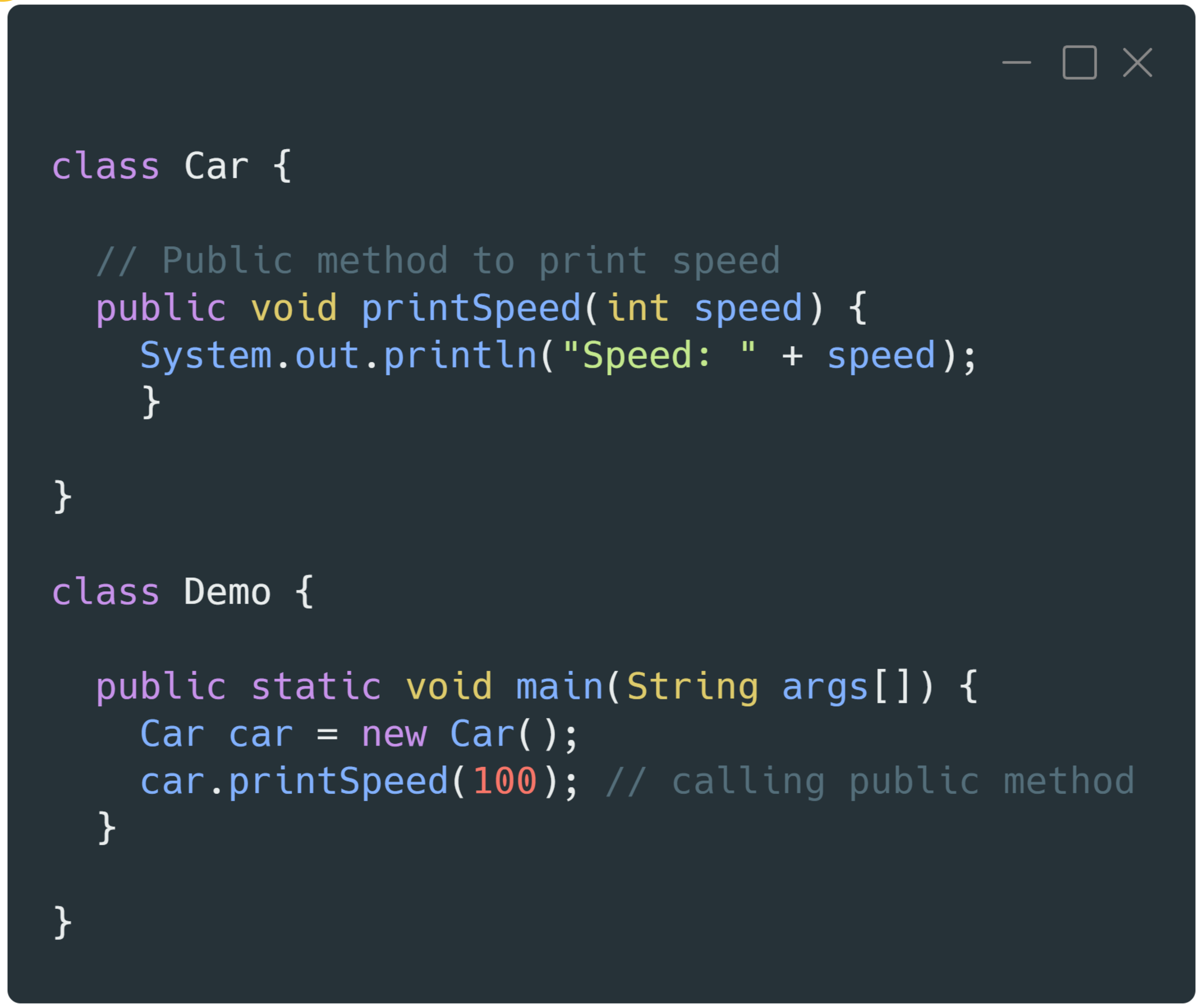
```
//Example of Initialization and Instantiation on the same line
SomeClass s; // Declaration
s = new SomeClass(); // Instantiates and initializes the memory and initializes the variable 's'
```

```
//Example of Initialization of a variable on a different line to memory
void someFunction(SomeClass other) {
    SomeClass s; // Declaration
    s = other; // Initializes the variable 's' but memory for variable "other" was set somewhere else
}
```

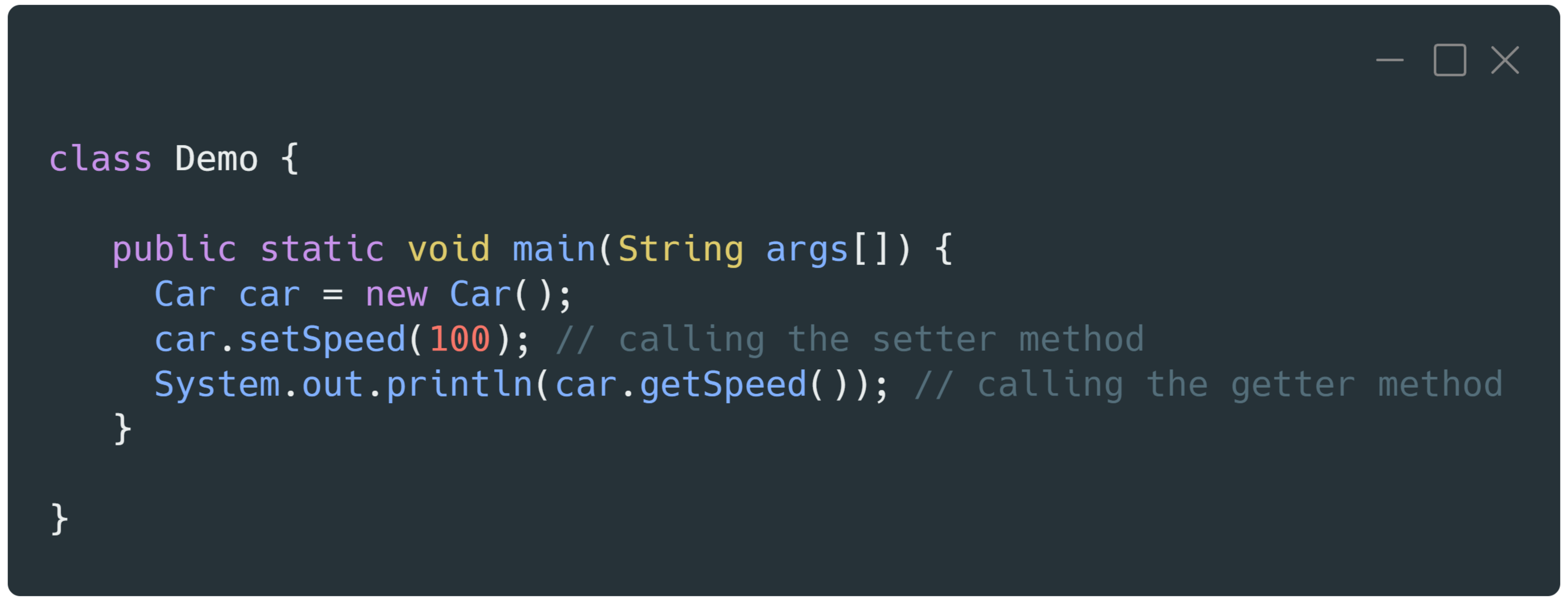
```
//The Structure of a Java Class
class Car { // Class name

  // Class Data members
  int topSpeed;
  int totalSeats;
  int fuelCapacity;
  String manufacturer;

  // Class Methods
  void refuel(){
    ...
  }
  void park(){
    ...
  }
  void drive(){
    ...
  }

  // Main method
  public static void main(String args[]) {
  ClassName obj = new ClassName(); // className object
}
```

OOP Building Blocks: Attributes & Methods

- **Attributes**
  - object's state
- **Methods**
  - object's behavior
  - Method parameters
  - Method Overloading
- **Constructor vs. Destructor**



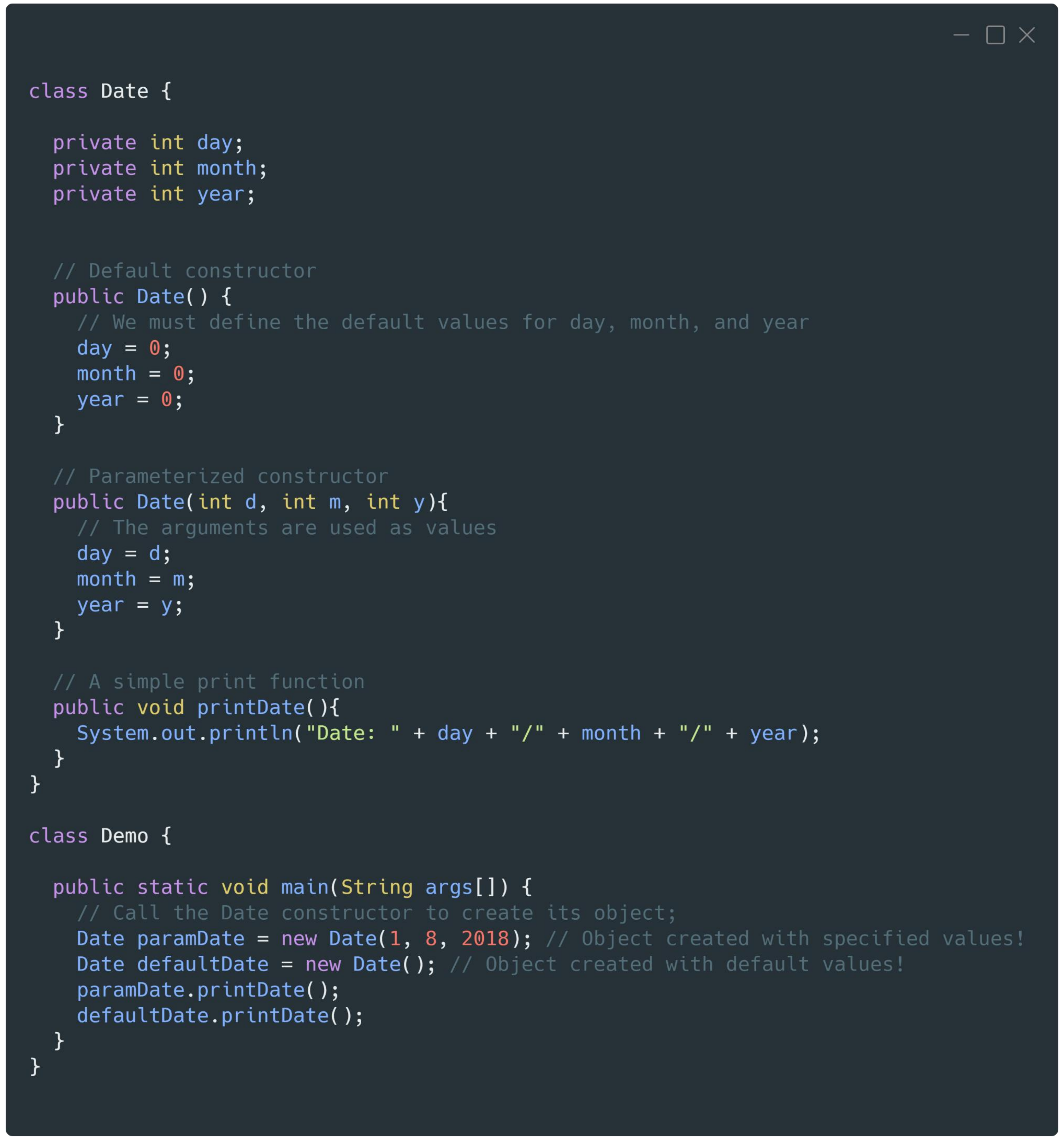
```java
class Car {

  // Public method to print speed
  public void printSpeed(int speed) {
    System.out.println("Speed: " + speed);
    }

}

class Demo {

  public static void main(String args[]) {
    Car car = new Car();
    car.printSpeed(100); // calling public method
  }

}
```



```java
class Demo {

   public static void main(String args[]) {
     Car car = new Car();
     car.setSpeed(100); // calling the setter method
     System.out.println(car.getSpeed()); // calling the getter method
   }

}
```

OOP Building Blocks: Constructors

- **Constructor**
  - Initializing new object states
  - Can be overloaded
  - No return type
  - Only called once
- **2 Types**
  - Default / non-parameterized
  - Parameterized



```java
class Date {

  private int day;
  private int month;
  private int year;


  // Default constructor
  public Date() {
    // We must define the default values for day, month, and year
    day = 0;
    month = 0;
    year = 0;
  }

  // Parameterized constructor
  public Date(int d, int m, int y){
    // The arguments are used as values
    day = d;
    month = m;
    year = y;
  }

  // A simple print function
  public void printDate(){
    System.out.println("Date: " + day + "/" + month + "/" + year);
  }
}

class Demo {

  public static void main(String args[]) {
    // Call the Date constructor to create its object;
    Date paramDate = new Date(1, 8, 2018); // Object created with specified values!
    Date defaultDate = new Date(); // Object created with default values!
    paramDate.printDate();
    defaultDate.printDate();
  }
}
```

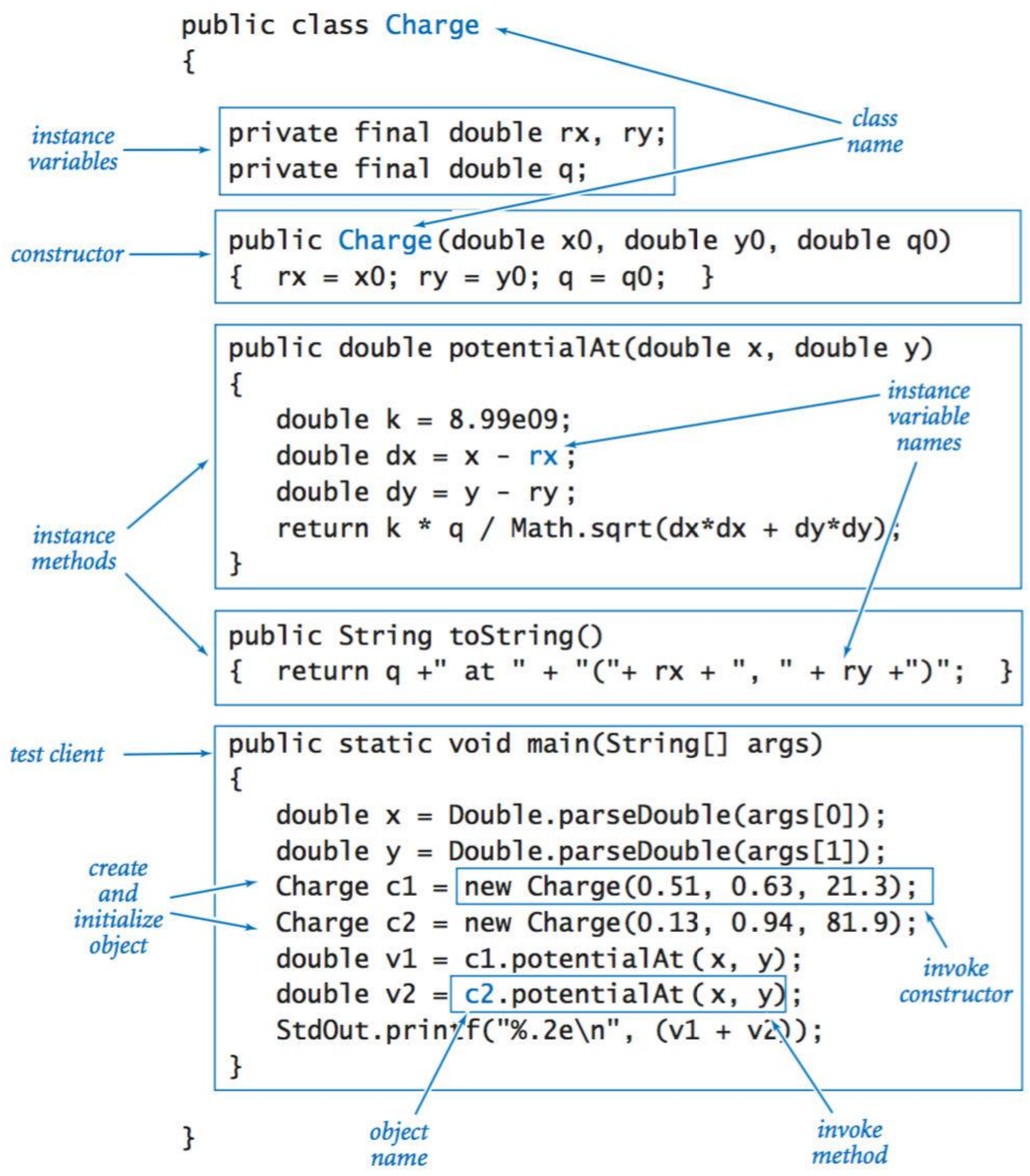
OOP Building Blocks: Constructors

```
A obA=new A();
```

```
class A
{
  A()
   {
     //some code
   }
}
```

```
class A
{
  public A()
  {
    System.out.println("Constructor with no parameter");
  }
  public A(int a)
  {
    System.out.println("Constructor with one integer parameter");
  }
  public A(int a,int b)
  {
    System.out.println("Constructor with two integer parameter");
  }
  public A(double a)
  {
    System.out.println("Constructor with one double parameter");
  }
}
```

Java OOP Cheat sheet



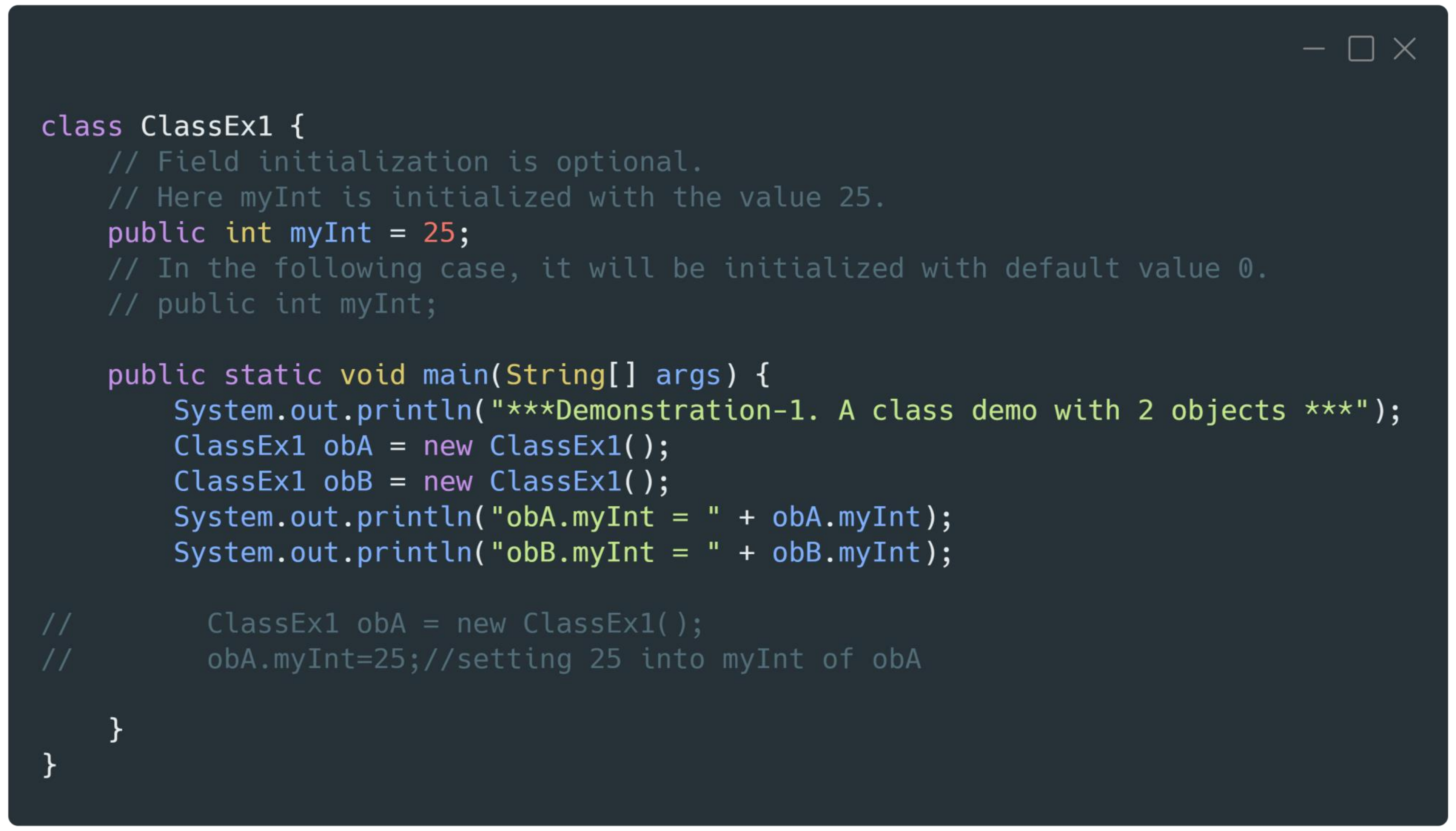
```
public class Charge
{
    private final double rx, ry;
    private final double q;

    public Charge(double x0, double y0, double q0)
    {   rx = x0; ry = y0; q = q0;   }

    public double potentialAt(double x, double y)
    {
        double k = 8.99e09;
        double dx = x - rx;
        double dy = y - ry;
        return k * q / Math.sqrt(dx*dx + dy*dy);
    }

    public String toString()
    {   return q +" at " + "("+ rx + ", " + ry +")";   }

    public static void main(String[] args)
    {
        double x = Double.parseDouble(args[0]);
        double y = Double.parseDouble(args[1]);
        Charge c1 = new Charge(0.51, 0.63, 21.3);
        Charge c2 = new Charge(0.13, 0.94, 81.9);
        double v1 = c1.potentialAt(x, y);
        double v2 = c2.potentialAt(x, y);
        StdOut.printf("%.2e\n", (v1 + v2));
    }

}
```

Note: Source Examples

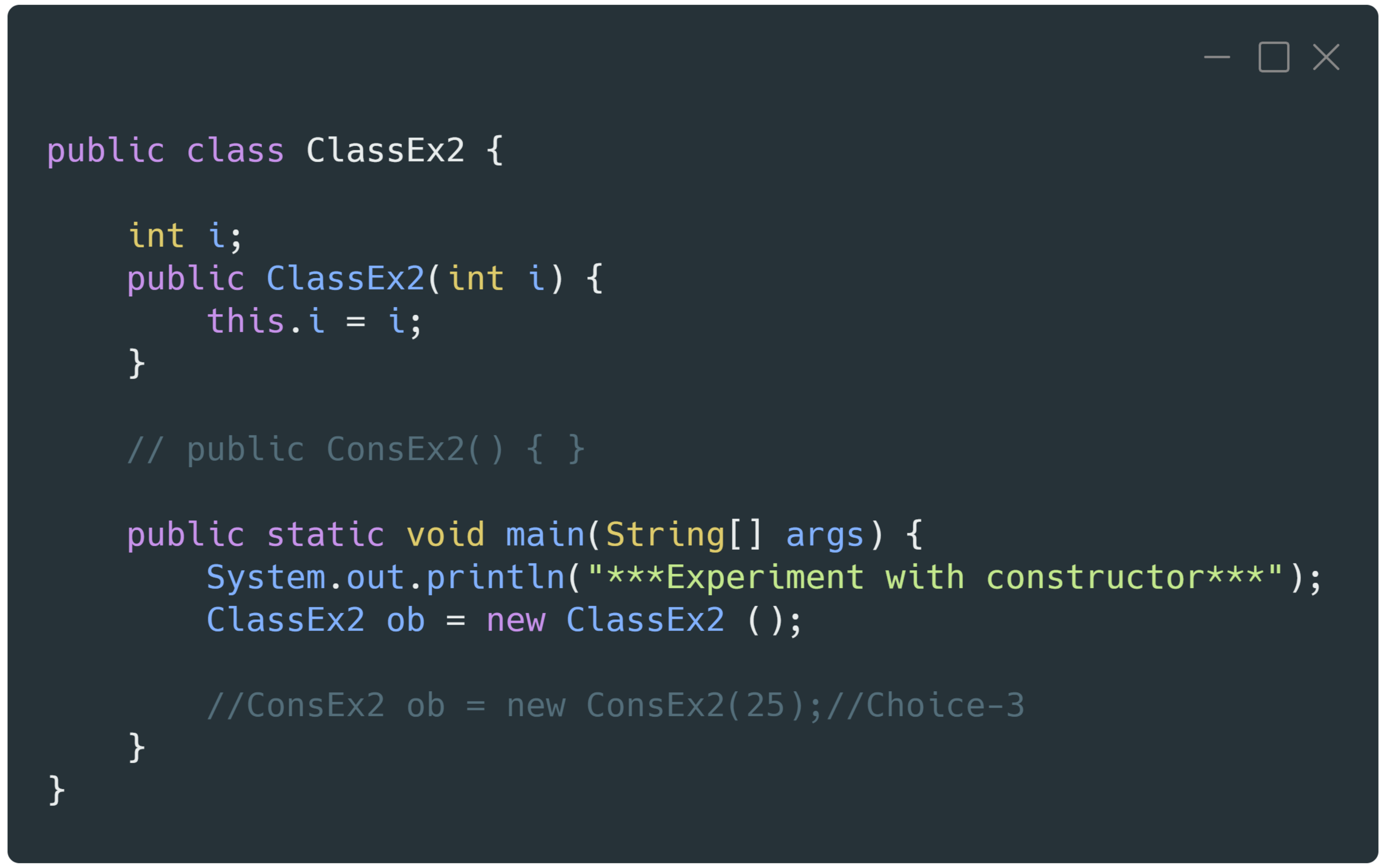
- **Source code examples that we are going through the lectures of this course are available at this repository:**
  - https://github.com/VahidAlizadeh/SE350Spring2021.git
  - Each session's codes will be pushed and available at the beginning of the session.

Examples (1 & 2)

- **Example Code 1** [package oopBasics1]
  - Q&A
    - The constructors do not have any return type. Is their return type "void?"
- **Example Code 2** [package oopBasics1]
  - Q&A
    - Why does the compiler give error? Shouldn't I have a default constructor?



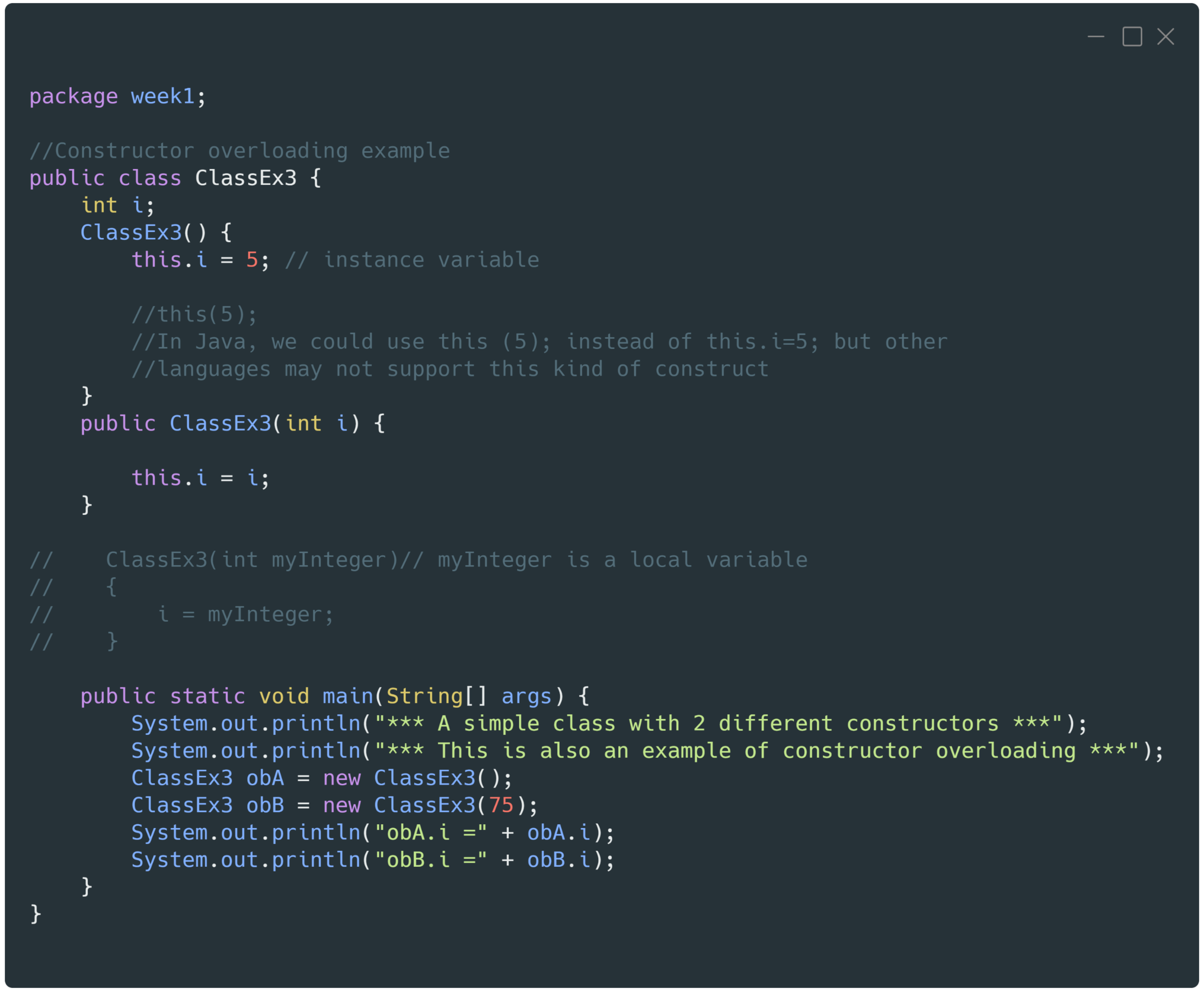
```java
class ClassEx1 {
    // Field initialization is optional.
    // Here myInt is initialized with the value 25.
    public int myInt = 25;
    // In the following case, it will be initialized with default value 0.
    // public int myInt;

    public static void main(String[] args) {
        System.out.println("***Demonstration-1. A class demo with 2 objects ***");
        ClassEx1 obA = new ClassEx1();
        ClassEx1 obB = new ClassEx1();
        System.out.println("obA.myInt = " + obA.myInt);
        System.out.println("obB.myInt = " + obB.myInt);

//        ClassEx1 obA = new ClassEx1();
//        obA.myInt=25;//setting 25 into myInt of obA

    }
}
```



```java
public class ClassEx2 {

    int i;
    public ClassEx2(int i) {
        this.i = i;
    }

    // public ConsEx2() { }

    public static void main(String[] args) {
        System.out.println("***Experiment with constructor***");
        ClassEx2 ob = new ClassEx2 ();

        //ConsEx2 ob = new ConsEx2(25);//Choice-3
    }
}
```

Examples (3)

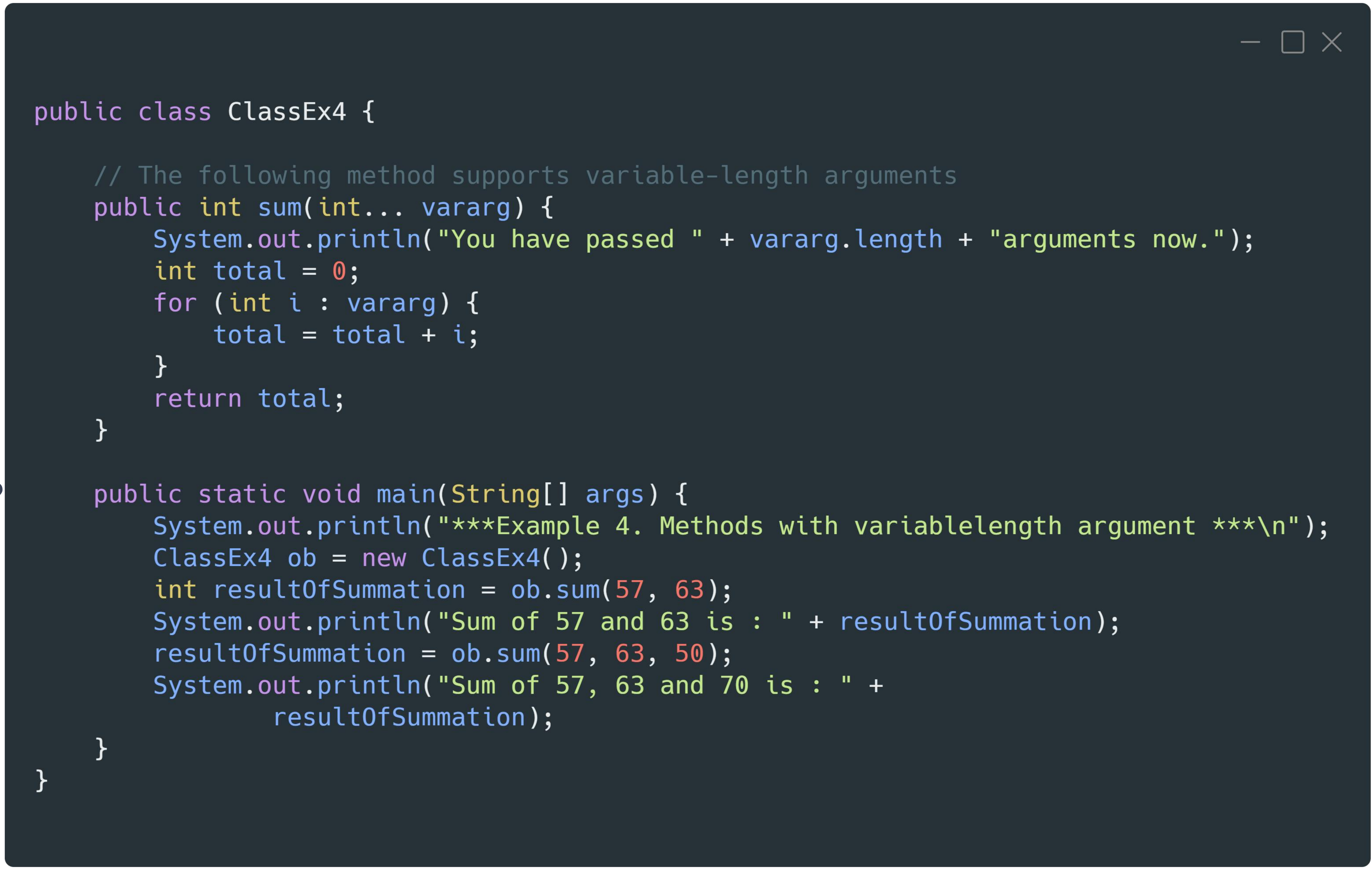
- **Example Code 3** [package oopBasics1]
  - Constructor **Overloading** example
  - Q&A
    - What is *this* keyword?
    - Local variable
      - Declared inside methods, blocks, or constructors
    - Instance variable
      - Declared inside a class but outside a method, block, or constructor



```java
package week1;

//Constructor overloading example
public class ClassEx3 {
    int i;
    ClassEx3() {
        this.i = 5; // instance variable

        //this(5);
        //In Java, we could use this (5); instead of this.i=5; but other
        //languages may not support this kind of construct
    }
    public ClassEx3(int i) {

        this.i = i;
    }

//    ClassEx3(int myInteger)// myInteger is a local variable
//    {
//        i = myInteger;
//    }

    public static void main(String[] args) {
        System.out.println("*** A simple class with 2 different constructors ***");
        System.out.println("*** This is also an example of constructor overloading ***");
        ClassEx3 obA = new ClassEx3();
        ClassEx3 obB = new ClassEx3(75);
        System.out.println("obA.i =" + obA.i);
        System.out.println("obB.i =" + obB.i);
    }
}
```

Examples (4)

- **Example Code 4** [package oopBasics1]
  - Q&A
    - Why **Vararg**?
    - Examples of some vararg methods in the Java library
      - printf() , format()
- **Benefits if OOP design in real-world scenarios?**
- **Summary**



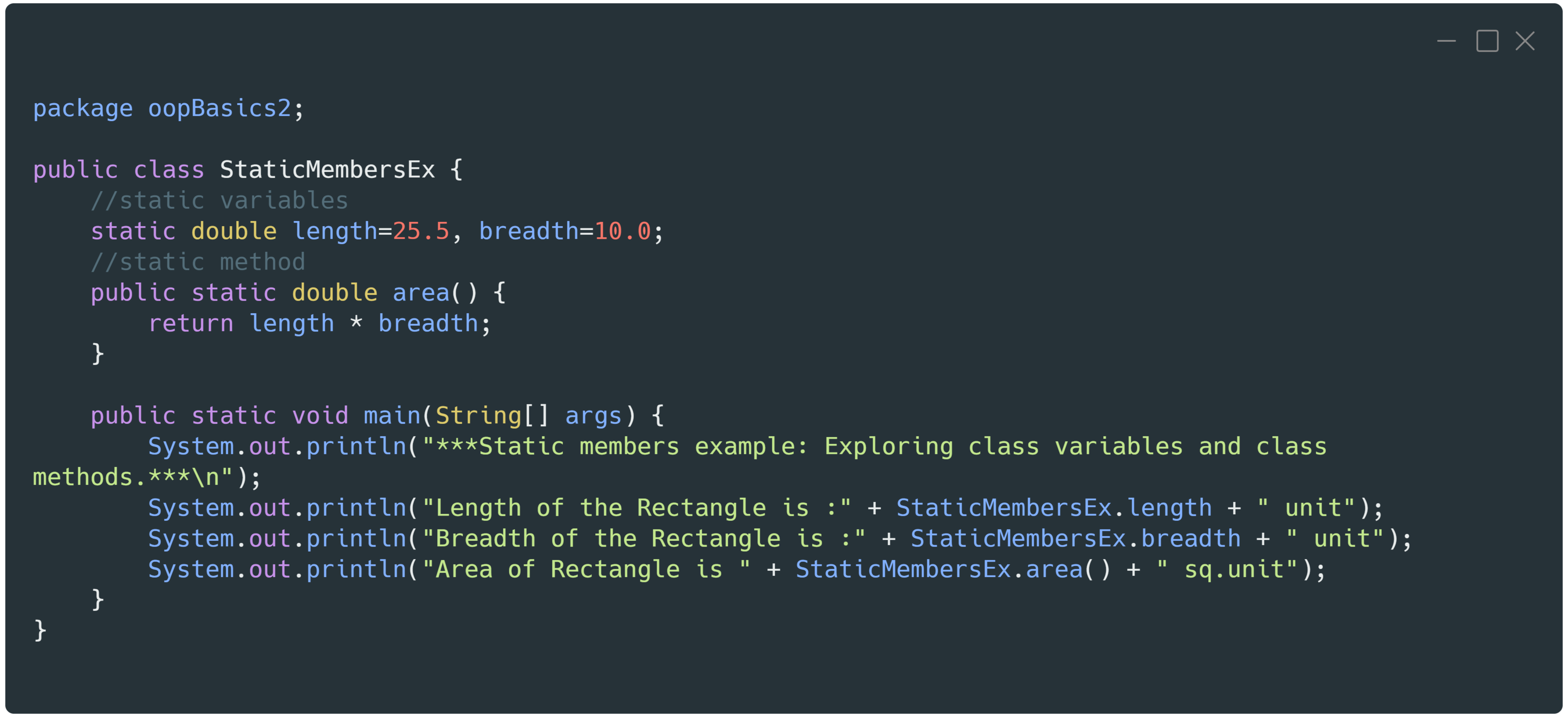
```java
public class ClassEx4 {

    // The following method supports variable-length arguments
    public int sum(int... vararg) {
        System.out.println("You have passed " + vararg.length + "arguments now.");
        int total = 0;
        for (int i : vararg) {
            total = total + i;
        }
        return total;
    }

    public static void main(String[] args) {
        System.out.println("***Example 4. Methods with variablelength argument ***\n");
        ClassEx4 ob = new ClassEx4();
        int resultOfSummation = ob.sum(57, 63);
        System.out.println("Sum of 57 and 63 is : " + resultOfSummation);
        resultOfSummation = ob.sum(57, 63, 50);
        System.out.println("Sum of 57, 63 and 70 is : " +
                resultOfSummation);
    }
}
```

Object-oriented Programming

Introduction & Basics 2: In-Depth

Static Methods and Variables

- **Class member vs. Instance member**
- **Static member = Class member**
  - Common to all instances of the class
  - ***Static*** keyword
- **Example** [package oopBasics2]
  - *package oopBasics2;*
  - Defining and accessing class members



```
package oopBasics2;

public class StaticMembersEx {
    //static variables
    static double length=25.5, breadth=10.0;
    //static method
    public static double area() {
        return length * breadth;
    }

    public static void main(String[] args) {
        System.out.println("***Static members example: Exploring class variables and class 
methods.***\n");
        System.out.println("Length of the Rectangle is :" + StaticMembersEx.length + " unit");
        System.out.println("Breadth of the Rectangle is :" + StaticMembersEx.breadth + " unit");
        System.out.println("Area of Rectangle is " + StaticMembersEx.area() + " sq.unit");
    }
}
```