DePaul University

OOP Intro and Basics (2)

Object-oriented Software Development
SE 350– Spring 2021

Vahid Alizadeh

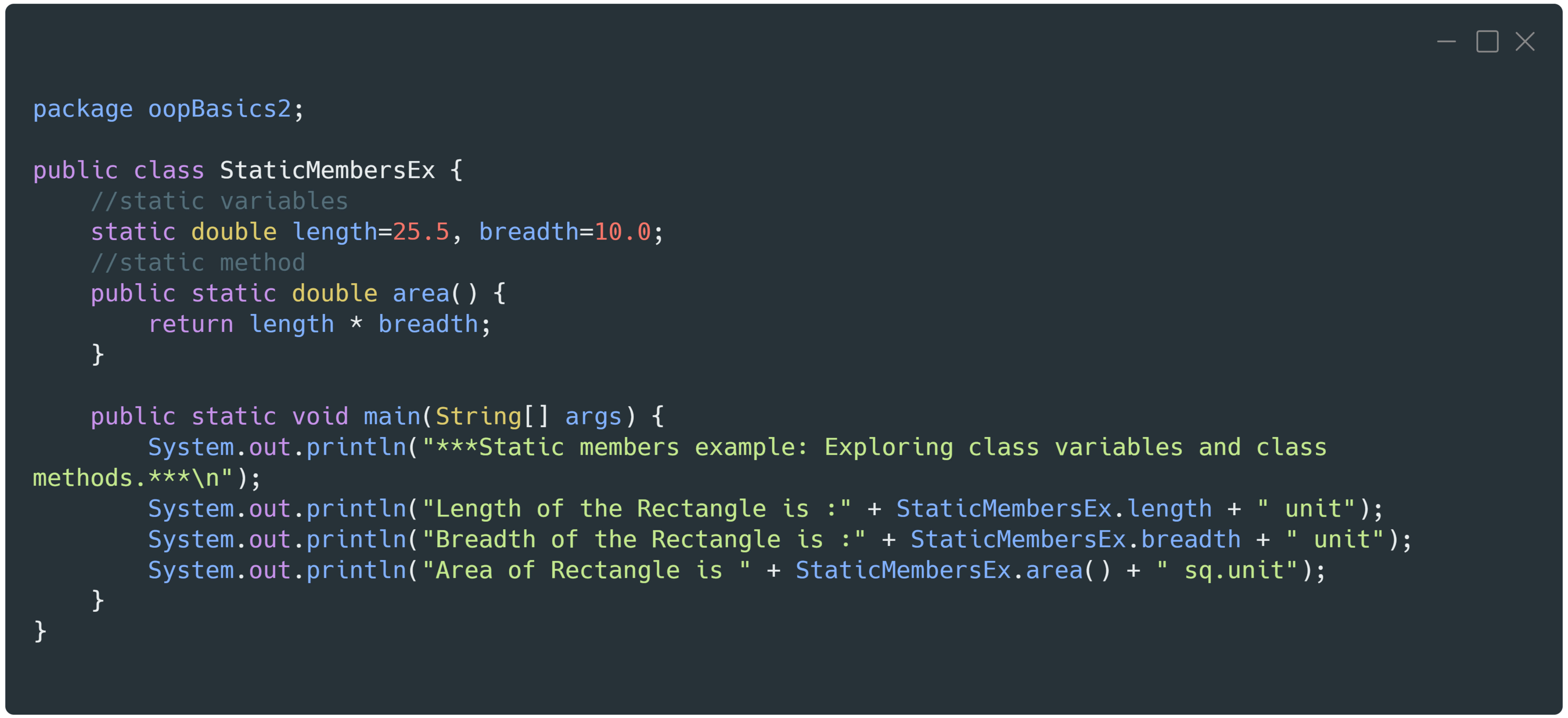
Week 2.2
April 8, 2021

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");
    }
}
```

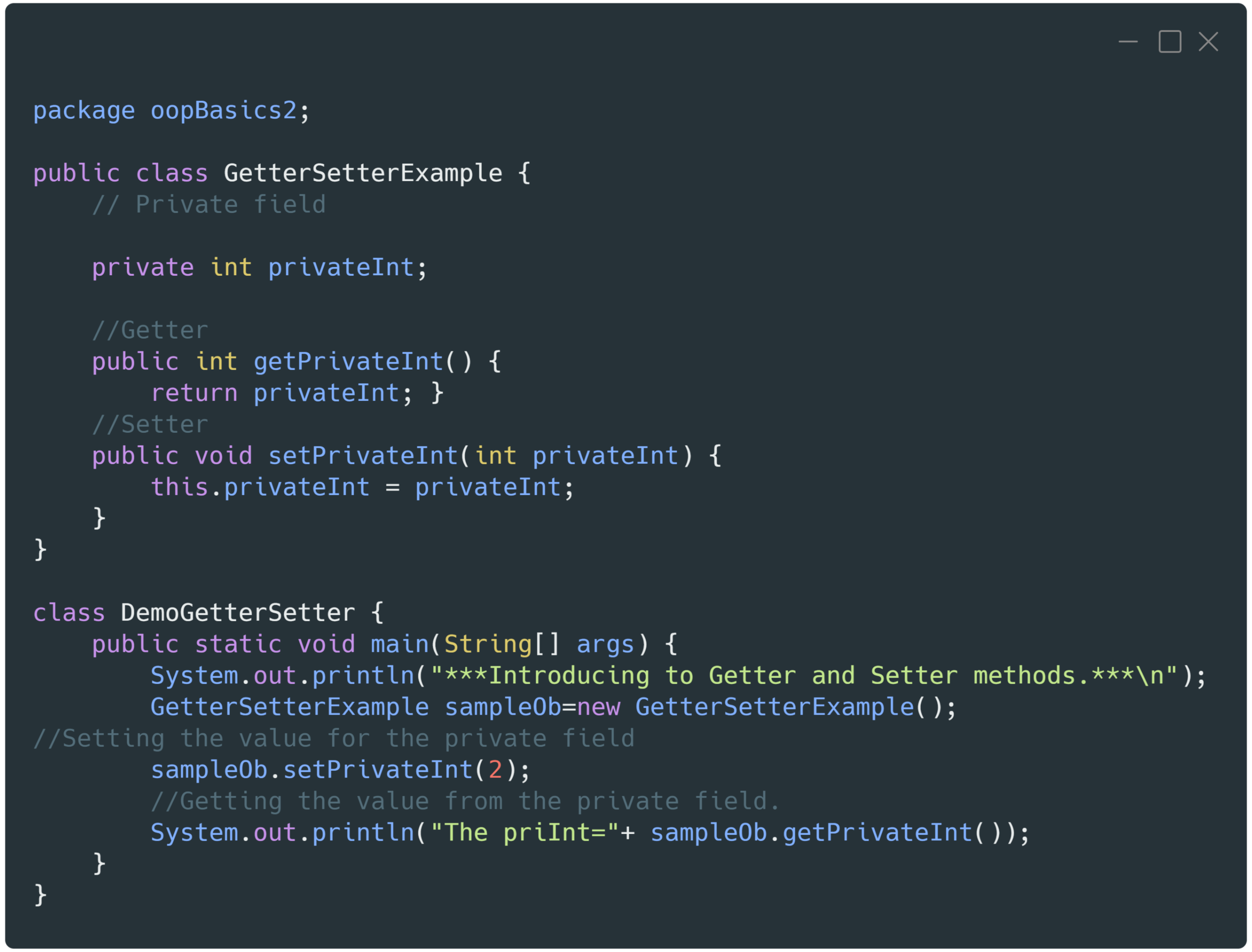
Access Control

- **Access modifiers**
  - Public, private, protected, default
  - public > protected > package-private (or default) > private
- ***Public* vs. *Private***
- **Why** *Main()* **method is always public?**
- **What happens when no access modifier is defined?**
- **Example** [package oopBasics2]
  - Accessing the public and private members
- **Levels of Access Control**
  1. **Class level** — Allowed modifiers: public, package-private (default).
  2. **Method level** — Allowed modifiers: public, private, protected, package-private (default).

```java
package oopBasics2;

public class AccessControlExample {
    // Public field
    public int publicInt = 1;

    // Public method
    public void showPublicMethod() {
        System.out.println("The showPublicMethod() is a public method.");
    }

    // Private field
    private int privateInt = 2;

    // Private method
    private void showPrivateMethod() {
        System.out.println("The showPrivateMethod() is a private method.");
    }
}

class Demo {
    public static void main(String[] args) {
        System.out.println("***Access control example. using private and public modifiers.***\n");
        AccessControlExample sampleOb = new AccessControlExample();
        System.out.println("The pubInt=" + sampleOb.publicInt);// 1
        sampleOb.showPublicMethod();
//      Compile-time error
//      System.out.println(" The priInt="+ sampleOb.privateInt);
//      Compile-time error
//      sampleOb.showPrivateMethod() ;
    }
}
```

Getters and Setters

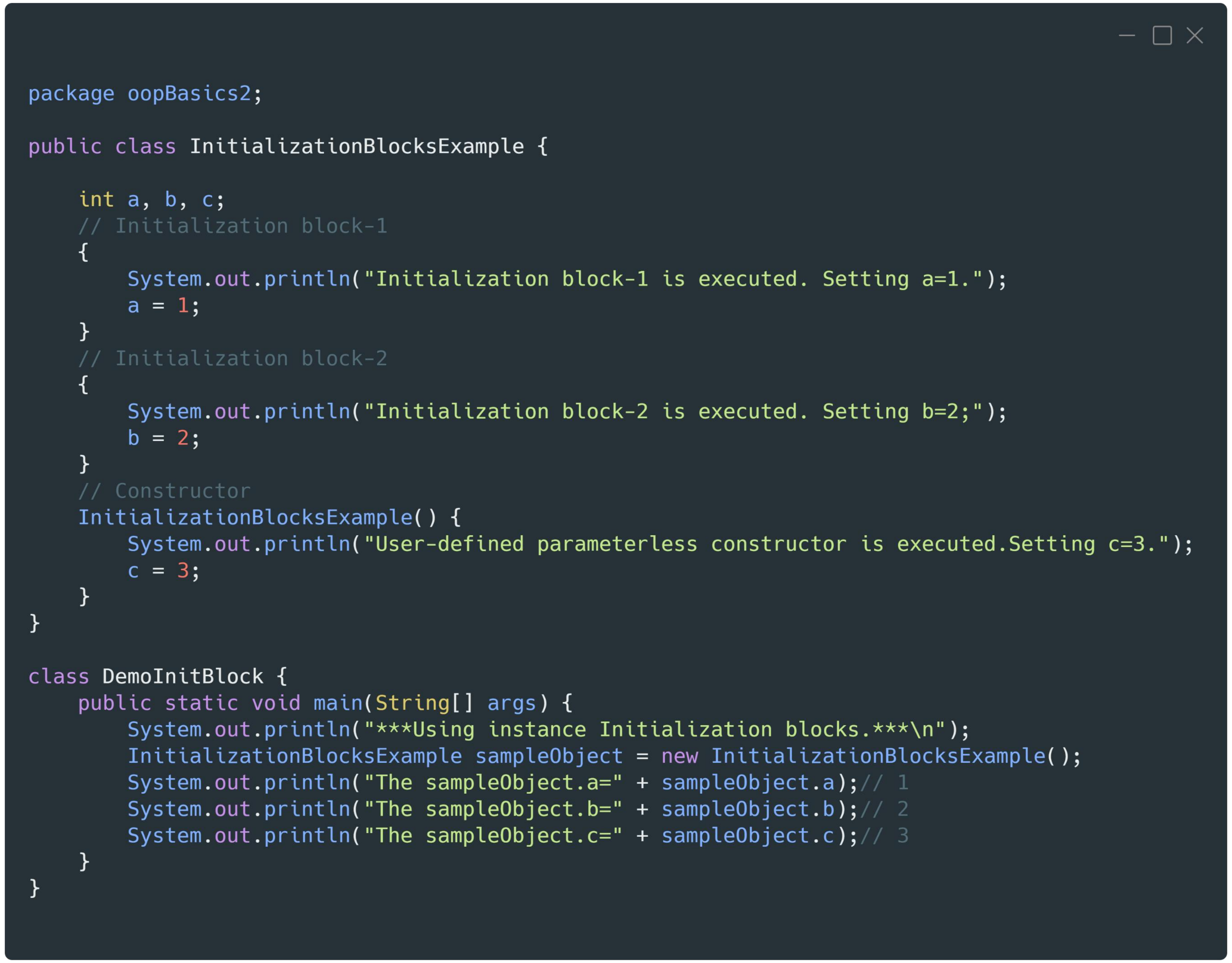
- **How to access private members of a class?**
  - Public getter and setter methods
- **Example** [package oopBasics2]
  - Creating getter and setter
  - Generate with IDE
  - Using Lombok Project
    - https://projectlombok.org/
- **Q&A**
  - What are the benefits of using Getter & Setter?
    - Controlled access
    - Read-only vs. Write-only
    - Increase data security



```java
package oopBasics2;

public class GetterSetterExample {
    // Private field

    private int privateInt;

    //Getter
    public int getPrivateInt() {
        return privateInt; }
    //Setter
    public void setPrivateInt(int privateInt) {
        this.privateInt = privateInt;
    }
}

class DemoGetterSetter {
    public static void main(String[] args) {
        System.out.println("***Introducing to Getter and Setter methods.***\n");
        GetterSetterExample sampleOb=new GetterSetterExample();
//Setting the value for the private field
        sampleOb.setPrivateInt(2);
        //Getting the value from the private field.
        System.out.println("The priInt="+ sampleOb.getPrivateInt());
    }
}
```

Initialization Block

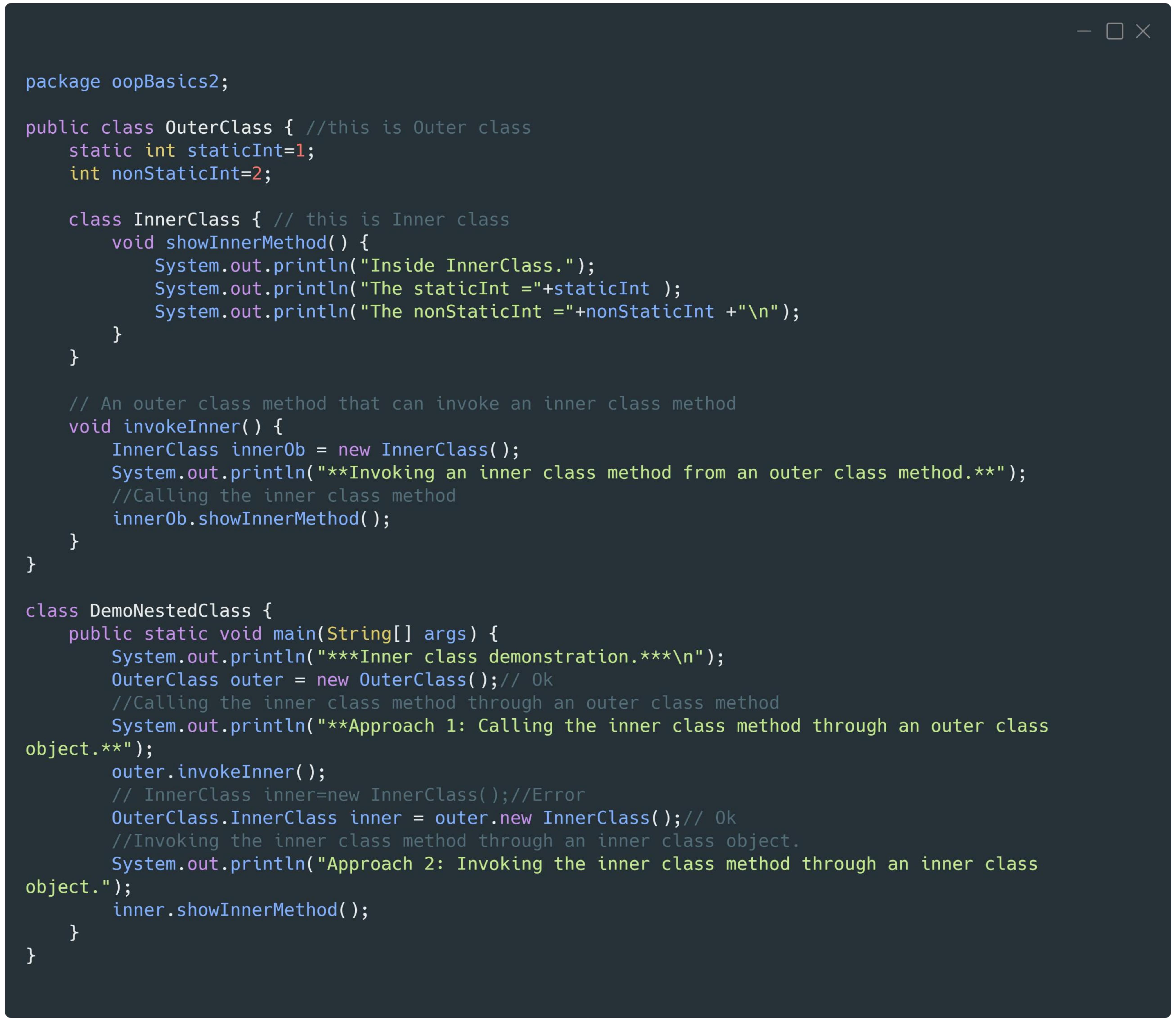
- **Initialization Blocks**
  - Alternatives to constructors
  - Can be static or non-static
  - Non-static initialization blocks = instance initialization blocks (IIB)
- **Example** [package oopBasics2]
  - Initialization block is called before constructor.
  - Initialization blocks are executed in the order of their appearance in your class.
- **Q&A**
  - Why would you need initialization block when you have constructors?
  - Is it possible to have multiple initialization blocks?



```java
package oopBasics2;

public class InitializationBlocksExample {

    int a, b, c;
    // Initialization block-1
    {
        System.out.println("Initialization block-1 is executed. Setting a=1.");
        a = 1;
    }
    // Initialization block-2
    {
        System.out.println("Initialization block-2 is executed. Setting b=2;");
        b = 2;
    }
    // Constructor
    InitializationBlocksExample() {
        System.out.println("User-defined parameterless constructor is executed.Setting c=3.");
        c = 3;
    }
}

class DemoInitBlock {
    public static void main(String[] args) {
        System.out.println("***Using instance Initialization blocks.***\n");
        InitializationBlocksExample sampleObject = new InitializationBlocksExample();
        System.out.println("The sampleObject.a=" + sampleObject.a);// 1
        System.out.println("The sampleObject.b=" + sampleObject.b);// 2
        System.out.println("The sampleObject.c=" + sampleObject.c);// 3
    }
}
```

Nested Class

- **Nested class**
  - One class inside another class
  - Can be static and non-static
  - Non-static nested class = inner class
- **Example** [package oopBasics2]
  - How to call an inner class?
    1. Via an outer class method
    2. Directly
- **Benefits of Nested Class**
  - Encapsulation with better security
  - Improve maintainability by grouping the classes logically



```java
package oopBasics2;

public class OuterClass { //this is Outer class
    static int staticInt=1;
    int nonStaticInt=2;

    class InnerClass { // this is Inner class
        void showInnerMethod() {
            System.out.println("Inside InnerClass.");
            System.out.println("The staticInt ="+staticInt );
            System.out.println("The nonStaticInt ="+nonStaticInt +"\n");
        }
    }

    // An outer class method that can invoke an inner class method
    void invokeInner() {
        InnerClass innerOb = new InnerClass();
        System.out.println("**Invoking an inner class method from an outer class method.**");
        //Calling the inner class method
        innerOb.showInnerMethod();
    }
}

class DemoNestedClass {
    public static void main(String[] args) {
        System.out.println("***Inner class demonstration.***\n");
        OuterClass outer = new OuterClass();// Ok
        //Calling the inner class method through an outer class method
        System.out.println("**Approach 1: Calling the inner class method through an outer class 
object.**");
        outer.invokeInner();
        // InnerClass inner=new InnerClass();//Error
        OuterClass.InnerClass inner = outer.new InnerClass();// Ok
        //Invoking the inner class method through an inner class object.
        System.out.println("Approach 2: Invoking the inner class method through an inner class 
object.");
        inner.showInnerMethod();
    }
}
```

Copying an Object

- **Creating new instance from scratch**
  - Costly, boring, time-consuming, and error-prone
  - Copy constructor, Object cloning, Serialization, …
- **Example** [package oopBasics2]
  - How to write your own copy constructor
- **Java does not support a default copy constructor.**
- **SUMMARY of OOP Intro 2…**

```java
class Student
{
    int studentID;
    String name;
    //Instance Constructor
    public Student(int studentID, String name)
    {
        this.studentID = studentID;
        this.name = name;
    }
    //Copy Constructor
    public Student( Student student)
    {
        this.name = student.name;
        this.studentID = student.studentID;
    }
    public void displayDetails()
    {
        System.out.println(" Student name: " + name + ",Student ID: "+ studentID);
    }
}

class DemoCopyConstructor {
    public static void main(String[] args) {
        System.out.println("***User-defined copy constructor***\n");
        Student student1 = new Student(123456, "Vahid");
        System.out.println(" The Student1 details:");
        student1.displayDetails();
        System.out.println("\n Copying student1 to student2 >>>");
        //Invoking the user-defined copy constructor
        Student student2 = new Student (student1);
        System.out.println(" The Student2 details:");
        student2.displayDetails();
    }
}
```

Class Activity

Get Ready!

Class Activity - Week2.2

Get ready to compete!

Q1: A class declared as "private" cannot be used outside its package.

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q1: A class declared as "private" cannot be used outside its package.

True

False

Total Results: 0

Q1: A class declared as "private" cannot be used outside its package.

Powered by Poll Everywhere

Leaderboard

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q2: This class declaration is incorrect: public static class MyClass {…}

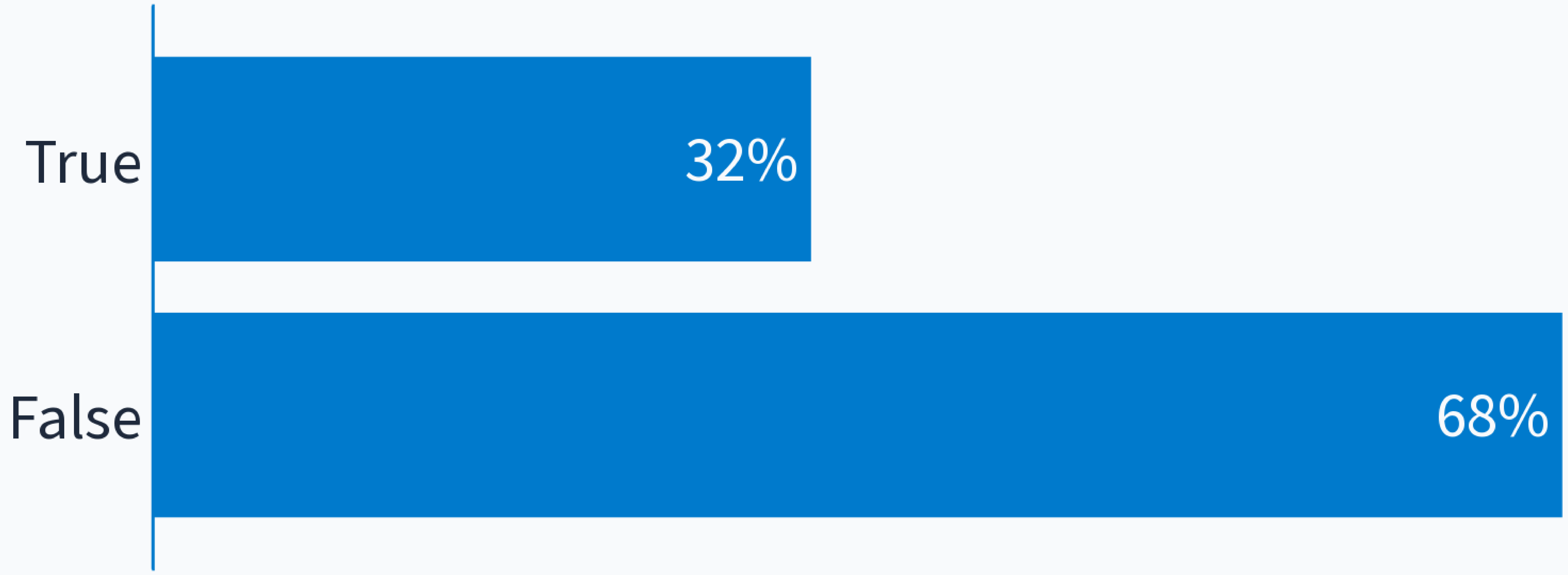
Q2: This class declaration is incorrect: public static class MyClass {...}

True

False

Total Results: 22

Q2: This class declaration is incorrect: public static class MyClass {...}



Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Leaderboard

Q3: This following class declaration is incorrect:

protected class MyClass {…}

Q3: This following class declaration is incorrect: protected class MyClass {...}

True

False

Total Results: 0

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q3: This following class declaration is incorrect: protected class MyClass {...}

True

False

Powered by Poll Everywhere

Leaderboard

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q4: This method declaration is correct: public void static MyMethod () {...}

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

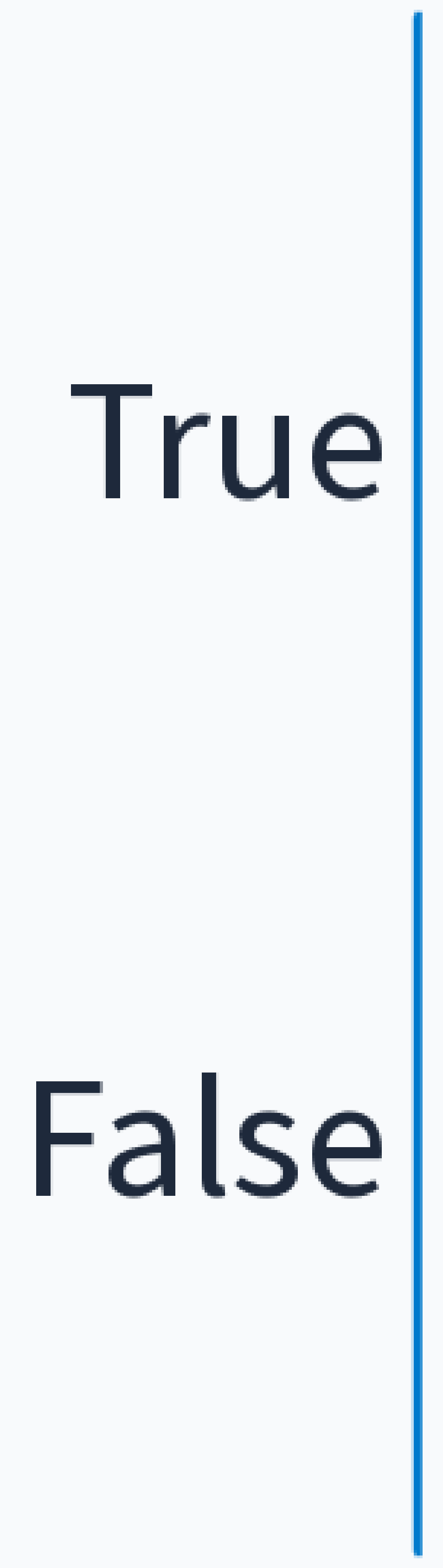
Q4: This method declaration is correct: public void static MyMethod () {...}

True

False

Total Results: 0

Q4: This method declaration is correct: public void static MyMethod () {...}



Powered by Poll Everywhere

Leaderboard

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q5: If a method is declared static, it cannot be called outside of its package.

Q5: If a method is declared static, it cannot be called outside of its package.

True

False

Total Results: 0

Powered by Poll Everywhere

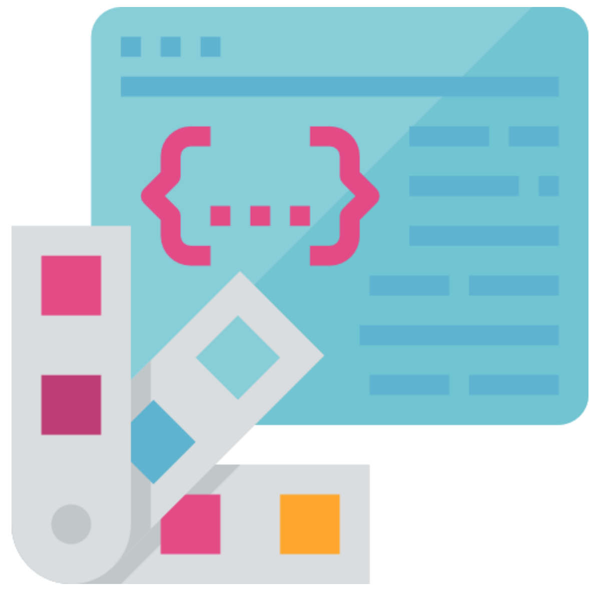
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q5: If a method is declared static, it cannot be called outside of its package.

Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Leaderboard

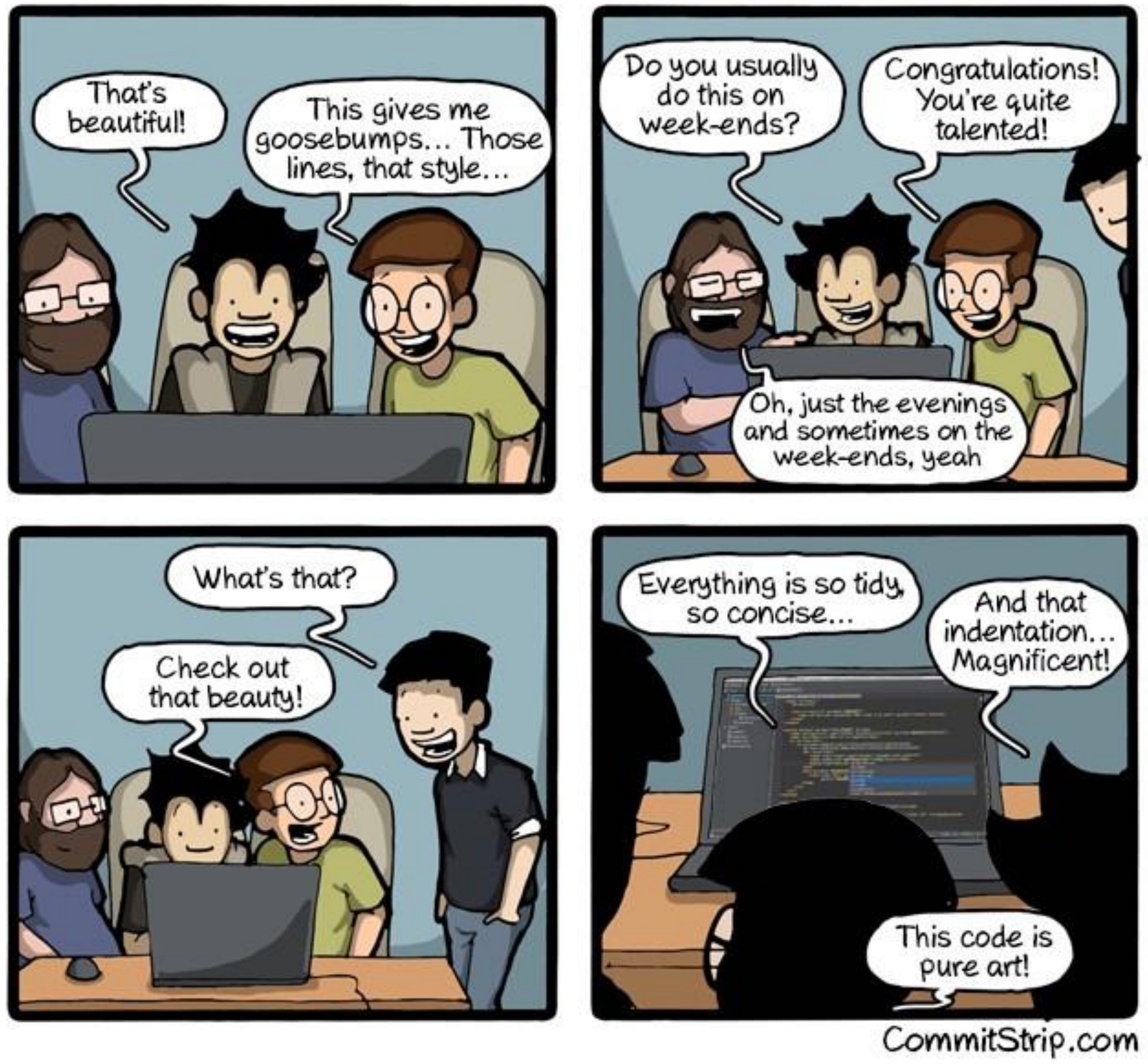


Code Conventions

For Java

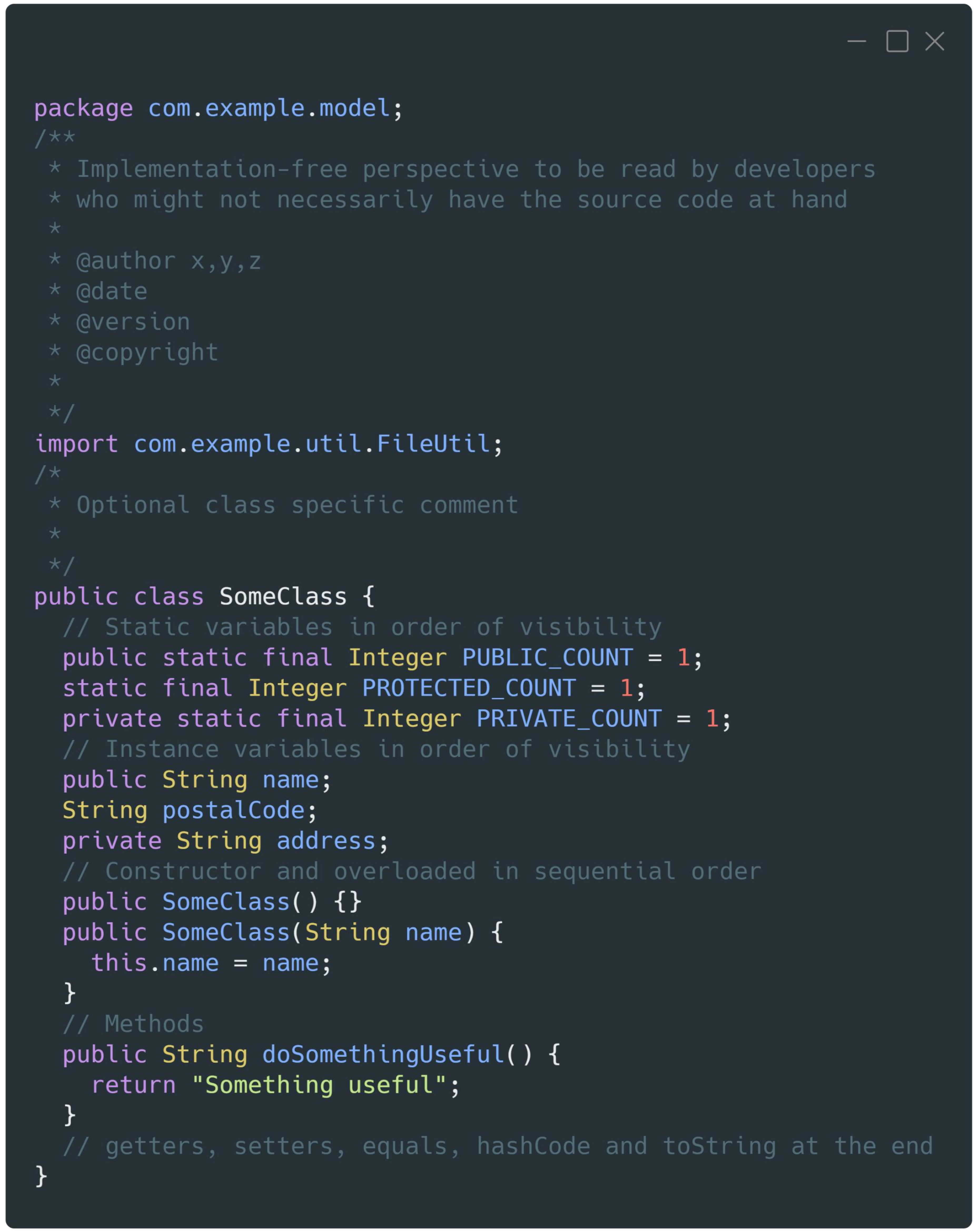
Java Code Conventions

- **What are coding conventions?**
  - Sets of guidelines: coding style, best practices, methods
- **What are the goals?**
  - Maintaining unified code style
  - Minimizing software maintenance costs
  - Improving software readability
  - Speeding up the work
- **Oracle Code Conventions for the Java Programming Language**
  - Last revision: 1999
- **Google Java Style Guide**
  - Last revision: 2018


That's beautiful!
This gives me goosebumps... Those lines, that style...
Do you usually do this on week-ends?
Congratulations! You're quite talented!
Oh, just the evenings and sometimes on the week-ends, yeah
What's that?
Check out that beauty!
Everything is so tidy, so concise...
And that indentation... Magnificent!
This code is pure art!
CommitStrip.com

Java Source File Conventions

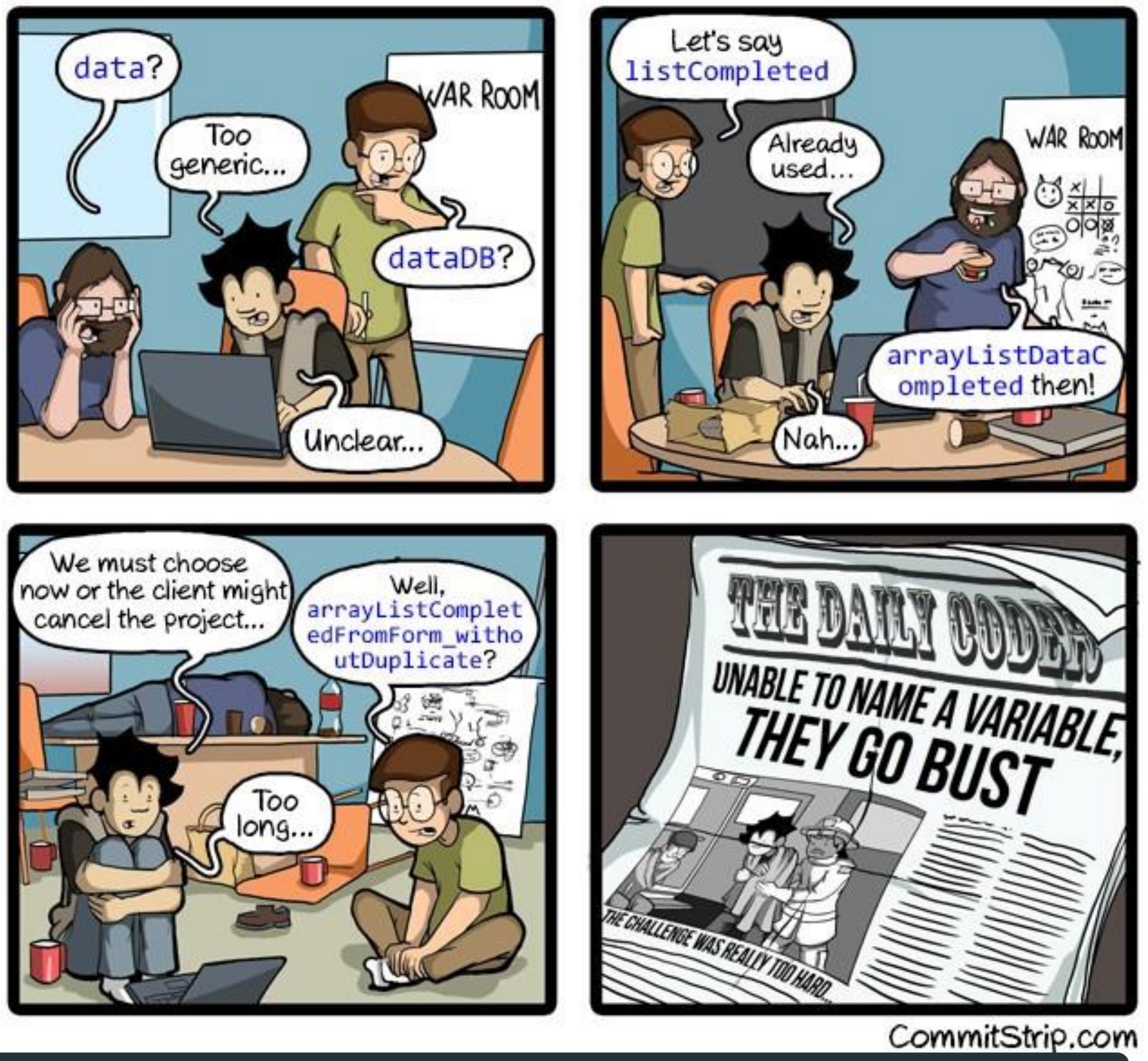
- **Best practices for java source file**
  - Length < 2000 LoC!
  - Source organization:
    - package declaration
    - documentation comment
    - imports
    - class comment
    - class/interface signature
    - Etc.

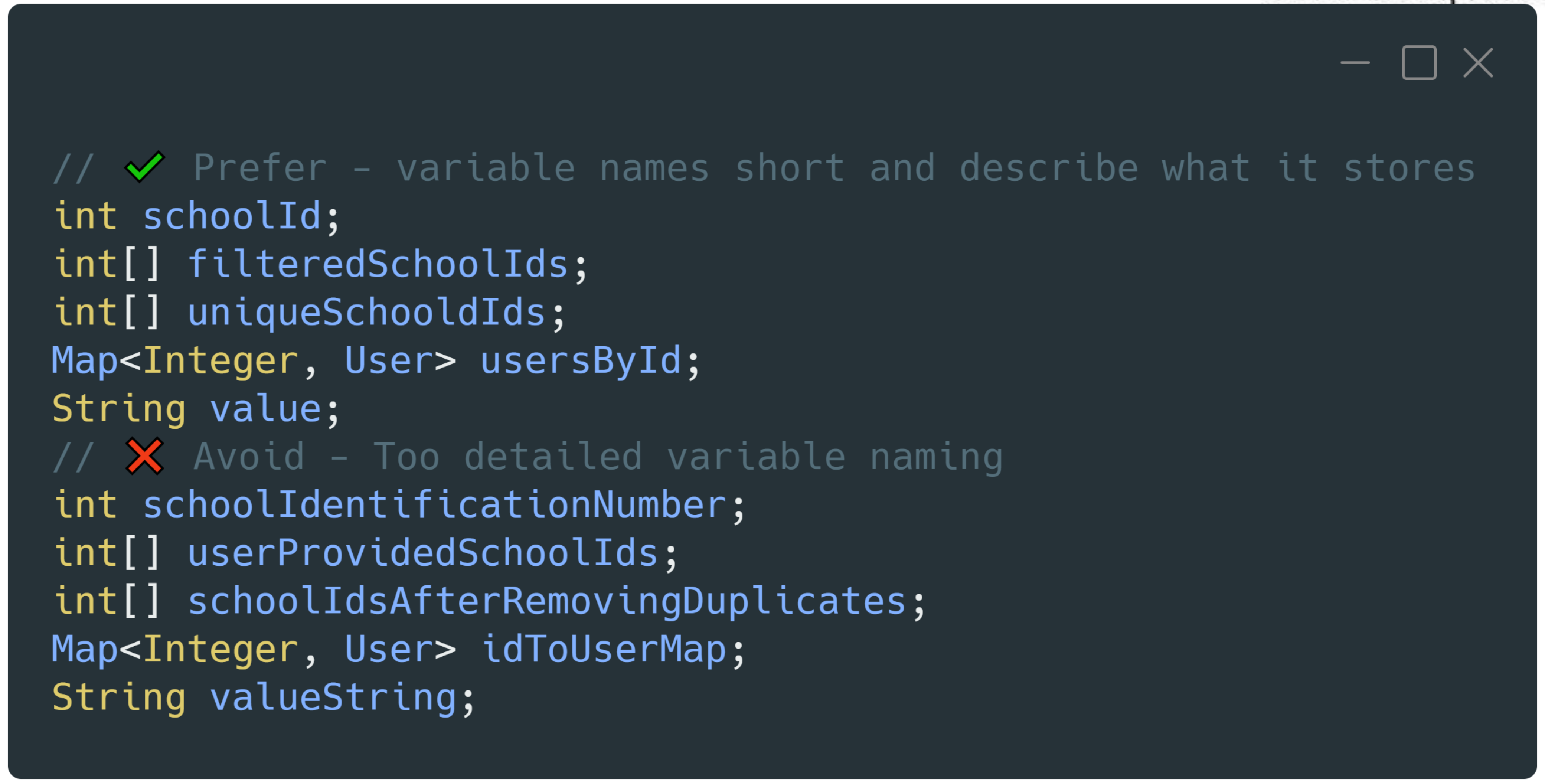


```java
package com.example.model;
/**
 * Implementation-free perspective to be read by developers
 * who might not necessarily have the source code at hand
 *
 * @author x,y,z
 * @date
 * @version
 * @copyright
 *
 */
import com.example.util.FileUtil;
/*
 * Optional class specific comment
 *
 */
public class SomeClass {
  // Static variables in order of visibility
  public static final Integer PUBLIC_COUNT = 1;
  static final Integer PROTECTED_COUNT = 1;
  private static final Integer PRIVATE_COUNT = 1;
  // Instance variables in order of visibility
  public String name;
  String postalCode;
  private String address;
  // Constructor and overloaded in sequential order
  public SomeClass() {}
  public SomeClass(String name) {
    this.name = name;
  }
  // Methods
  public String doSomethingUseful() {
    return "Something useful";
  }
  // getters, setters, equals, hashCode and toString at the end
}
```

Naming Convention

- **Set of rules to make the code look uniform.**
- **Package**
  - Lowercase. Ex: java.lang
- **Class, enum, interface, and annotation**
  - Uppercase. Ex: Thread, Runnable, @Override
- **Methods and field names**
  - camel case. Ex: add, isNull
- **Constants**
  - Uppercase + underscore. Ex: MIN_VALUE
- **Local variables**
  - camel case
- **File**
  - CamelCase - matching the class name


data?
Too generic...
WAR ROOM
dataDB?
Unclear...
Let's say listCompleted
Already used...
WAR ROOM
arrayListDataCompleted then!
Nah...
We must choose now or the client might cancel the project...
Well, arrayListCompletedFromForm_withoutDuplicate?
Too long...
THE DAILY CODER
UNABLE TO NAME A VARIABLE, THEY GO BUST
THE CHALLENGE WAS REALLY TOO HARD
CommitStrip.com



```
// ✓ Prefer - variable names short and describe what it stores
int schoolId;
int[] filteredSchoolIds;
int[] uniqueSchoolIds;
Map<Integer, User> usersById;
String value;
// ✗ Avoid - Too detailed variable naming
int schoolIdentificationNumber;
int[] userProvidedSchoolIds;
int[] schoolIdsAfterRemovingDuplicates;
Map<Integer, User> idToUserMap;
String valueString;
```