

David Christos

Deborah Harding

CS2050

22 January 2025

## Technical Doc 1/22

Control Flow Statements: In general aspect a CFS is a type of code that is used like a sign wether to for example Rerun the code or jump to a different line of code as it can be used as a different way as a loop statement and or a if statement

Pass by Value: a placeholder for future code Example



Pass By References: a swap by using different integers

## M01 L01 notes

Control Flow Statements: Control flow statements are instructions that tell the computer how to run your program. They help the program make decisions, repeat actions, or jump to specific parts of the code.

## Different Types of Control Statements

1. If Statement
2. If-else statement

3. if -elif-else statement

#### Looping Statements

1. for loop
2. while loop

#### Jump Statements

1. break: Stops the loop early
2. Continue: skips to next round of the loop
3. Return: Ends a function and can send back a result

#### Pass by Value and Pass by Reference:

In programming, pass by value means a copy of the variable is sent to a function. Any changes made inside the function only affect the copy and not the original variable. This makes it safe because the original data stays the same.

Tec Doc 1/29/25

#### Chars are different then integers

Strings are arrays/ String of chars are just a array of character. String have a char of index located in the array which is basically a string which must stay in range,

- 1.Strings are stored in characters in the index(0)
- 2.they can be accessed by using the index such as starting STRING[]
3. It is used to put it in reverse and process each character by itself
- 4.since the index is starting at 0
- 5.all vowels will be lower case

# ARRAYS

## *HOW ARE ARRAYS PASSED*

By reference

Rules for array,

# SCOPE

What is Scope: Where a variable can be used and accessed

When you have methods in a class and use this you can reference to a object which is either hidden or in a different class

# ARRAY OF OBJECTS

Array of dogs,

For the dogs we hold the address and space for it, index 0 will be used for dog 1

GIST, dog 1 has the index of 0 because it stored at the address and has the address for it. This is important to allow the memory to be saved on the stack as either a NULL or allowing the space to be saved

M1 L3Tec Doc

DRY: DO NOT REPEAT YOURSELF

SRP:

We use UML to guide us

UML Means private plus public

Aggregation in a special form is when there is a owner

Student "has-a" Name

Student "has-a" Address

**Design A ZOO-EXAMPLE O**

```
class Dog { 4 usages new *
    static int numDogs = 0; // Shared by all Dog objects
    static void showCount() { no usages new *
        |   System.out.println("Total Dogs: " + numDogs);
        |   }
    }
}
Dog.showCount(); // Calling static method new *
```

**F DRY**

```

public class AnimalPolymorphism
{
    public static void main(String[] args)
    {
        System.out.println("Super Cool Polymorphism with Dynamic Binding Example");
        // Create an array to hold Animal objects
        AnimalP[] animals = new AnimalP[4];
        animals[0] = new HippoP();
        animals[1] = new LionP();
        animals[2] = new TigerP();
        animals[3] = new WolfP();

        VetP animalDoc = new VetP();

        // Now let's demonstrate the power of polymorphism!
        for (int i = 0; i < animals.length; i++)
        {
            System.out.println();
            System.out.print("Animal [" + i + "] is a ");
            if(animals[i] instanceof HippoP)
            {
                System.out.print("hippo\n");
            }else if(animals[i] instanceof LionP)
            {
                System.out.print("lion\n");
            }else if(animals[i] instanceof TigerP)
            {
                System.out.print("Tiger\n");
            }else
            {
                System.out.print("Wolf\n");
            }

            animals[i].eat();
            animals[i].sleep();
            System.out.print("Vet is giving a shot.");
            animalDoc.giveShot(animals[i]);
        }
    }
}

```

Why do they design this to avoid repetition

By using inheritance by placing the common code in a common class and then we extend the common class which can be use as a “IS-A” instead of a “HAS-A”

Making a superclass for subclass to pull the differences

Making a private class is using encapsulation

KEYWORDS: INHERITANCE, SUPERCLASS, SUBCLASS, METHOD OVERRIDING,  
METHOD OVERLOADING

Superclass Constructor

RESET

## Polymorphism:

Polymorphism in programming allows objects of different classes to be treated as instances of the same class through a shared interface. It enables methods to have the same name but different implementations based on the object calling them, promoting flexibility and code reusability. Polymorphism is commonly achieved through method overriding in inheritance and method overloading within the same class. Example of parent class and sub class

```
public class Christos6E01Polymorphism {  davidchristos
    public static void main(String[] args) {  davidchristos
        try {
            File file = new File( pathname: "/Users/davidchristos/IdeaProje
            Scanner scanner = new Scanner(file);

            while (scanner.hasNextLine()) {
                System.out.println(scanner.nextLine());
            }

            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found! Check the path.");
            e.printStackTrace();
        }
    }
}

// Superclass: Animal
class Animal {  4 usages  4 inheritors  davidchristos *
    private String name;  3 usages
    private String food;  2 usages
    private int weight;  3 usages
```

Classes and Objects: A **class** is a blueprint for creating objects, while an **object** is an instance of a class with specific values and behaviors.

Implement Encapsulation: Public and private classes

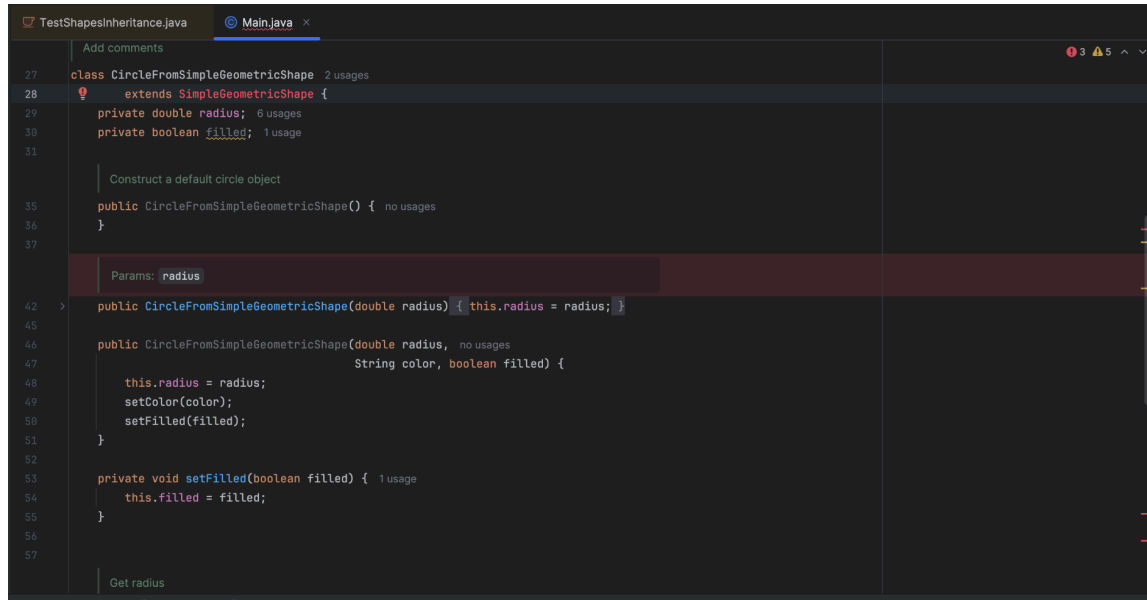
Static and Instance variables and methods: Static variables and methods belong to the class and are the same for all objects, while instance variables and methods belong to each object separately and can have different values.

Use constructors correctly, calling `setLength()` and `setWidth()` to validate inputs.

Ensures invalid values default to 1.0 to maintain logical consistency.

Provide getter and setter methods for both length and width.

Use proper method structure for calculating area and perimeter.



The screenshot shows an IDE with two tabs: 'TestShapesInheritance.java' and 'Main.java'. The 'Main.java' tab is active, displaying the following Java code:

```
27 class CircleFromSimpleGeometricShape 2 usages
28     extends SimpleGeometricShape {
29         private double radius; 6 usages
30         private boolean filled; 1 usage
31
32         Construct a default circle object
33
34         public CircleFromSimpleGeometricShape() { no usages
35         }
36
37
38 Params: radius
39
40 > public CircleFromSimpleGeometricShape(double radius) { this.radius = radius; }
41
42
43 public CircleFromSimpleGeometricShape(double radius, no usages
44                                     String color, boolean filled) {
45     this.radius = radius;
46     setColor(color);
47     setFilled(filled);
48 }
49
50 private void setFilled(boolean filled) { 1 usage
51     this.filled = filled;
52 }
53
54
55 Get radius
```

The code defines a class `CircleFromSimpleGeometricShape` that extends `SimpleGeometricShape`. It includes private fields `radius` and `filled`, a default constructor, a constructor with parameters `radius`, `color`, and `filled`, and a `setFilled` method. A parameter hint for `radius` is shown above the constructor with the parameter name `radius`. A `Get radius` hint is shown at the bottom.