# CPSC-224 Software Development

# Types and Objects

**Yu Wang**

**wangy2@gonzaga.edu**

**Jan 27, 2025**

# Announcement

- Quiz will be on Canvas
- Homework1 (Check the updates on Canvas)
  - CPSC 224 - GitHub Classroom Assignment Instructions
  - Deadline for submission: Feb. 12

# Review - Last Class

GONZAGA UNIVERSITY

- ✓ We learned - Strings and few important methods
- ✓ We learned - To name our classes, we use the PascalNamingConvention. To name our methods, we use the camelNamingConvention.

# Common Escape Sequences

GONZAGA UNIVERSITY

- In Java, **escape sequences** are special character combinations starting with a backslash (\) that represent characters that cannot be directly included in a string.
- These sequences allow you to include special characters like newlines, tabs, quotes, and more.

# Common Escape Sequences

| Escape Sequence | Meaning | Example |
|---|---|---|
| \n | Newline | "Hello\nWorld" → Hello<br>World |
| \t | Tab | "Hello\tWorld" → Hello World |
| \' | Single quote | "It\'s Java" → It's Java |
| \" | Double quote | "She said \"Hi\"" → She said "Hi" |
| \\ | Backslash | "C:\\Program Files" → C:\Program Files |
| \r | Carriage return | "Hello\rWorld" (rarely used) →World |
| \b | Backspace | "Java\bScript" → JavScript (removes the character before it) |

# Arrays

- We use arrays to store a list of items: like a list of numbers, or a list of people or a list of messages.
- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- Arrays are a fixed-size, indexed collection, where each element is accessed using its index starting from zero.

# Arrays and Examples

- To declare an array, define the variable type with square brackets: `String[] cars;`
- We have now declared a variable that holds an array of strings. To insert values to it, you can place the values in a comma-separated list, inside curly braces:

```
String [] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

# Arrays and Examples

- To create an array of integers, you could write:

```
int[] myNum = {10,20,30,40};
```

- You can access an array element by referring to the index number. Array indexes start with 0: [0] is the first element. [1] is the second element, etc.

```
String [] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
```

# Arrays and Examples

GONZAGA UNIVERSITY

- To find out how many elements an array has, use the length property:

```
String [] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
```

# Multidimensional Arrays

- A multidimensional array is an array of arrays. Multidimensional arrays are useful when you want to store data as a table (with rows and columns).
- To create a two-dimensional array, add each array within its own set of curly braces:

```
int[] [] myNum ={{1,2},{3,4,5}};
```

# Access Elements

- To access the elements of the myNum array, specify two indexes: one for the array, and one for the element inside that array.

```
int[] [] myNum ={{1,2},{3,4,5}};
System.out.println(myNum[1][2]);
```

# Others

- **Constant:** is a variable whose value cannot be changed after it is initialized. Constants are typically declared using the final keyword.
- **Casting** in Java refers to converting one data type into another. It is a way to tell the compiler to treat a value of one data type as another type.

# Casting Type

- **Widening Casting** (automatically) - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double

- **Narrowing Casting** (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

# Classes

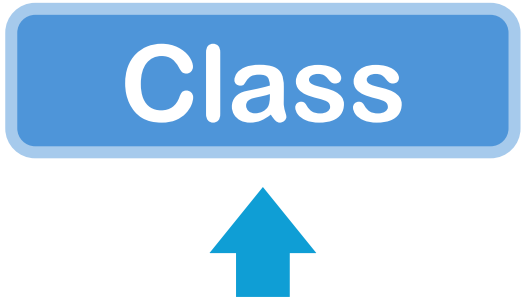
GONZAGA UNIVERSITY

## Classes

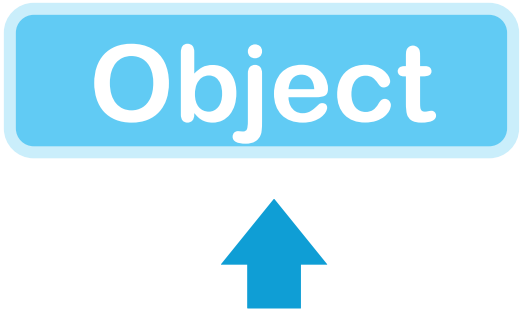
# Principles

- Encapsulation
- Abstraction
- Constructors
- Getters / Setters
- Method Overloading

# Class and Object

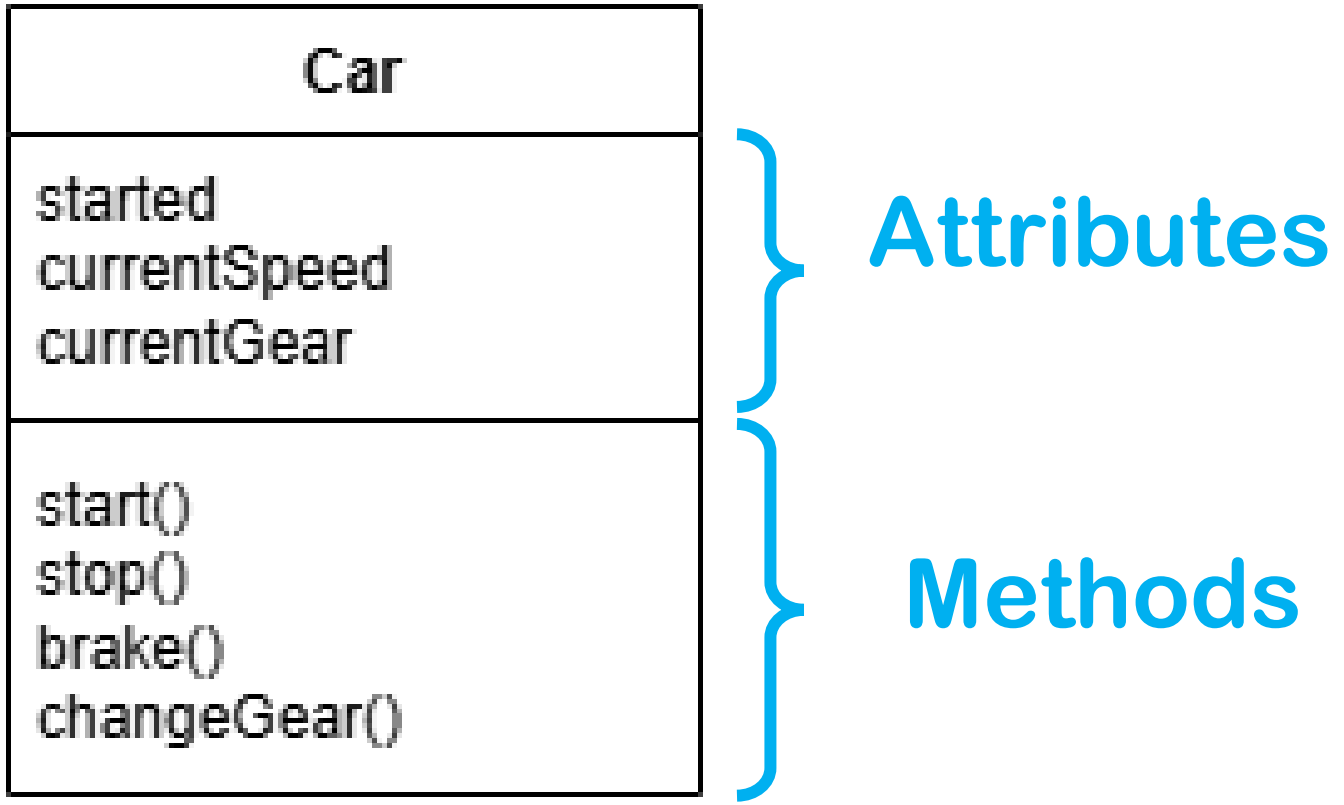



A blueprint for creating objects

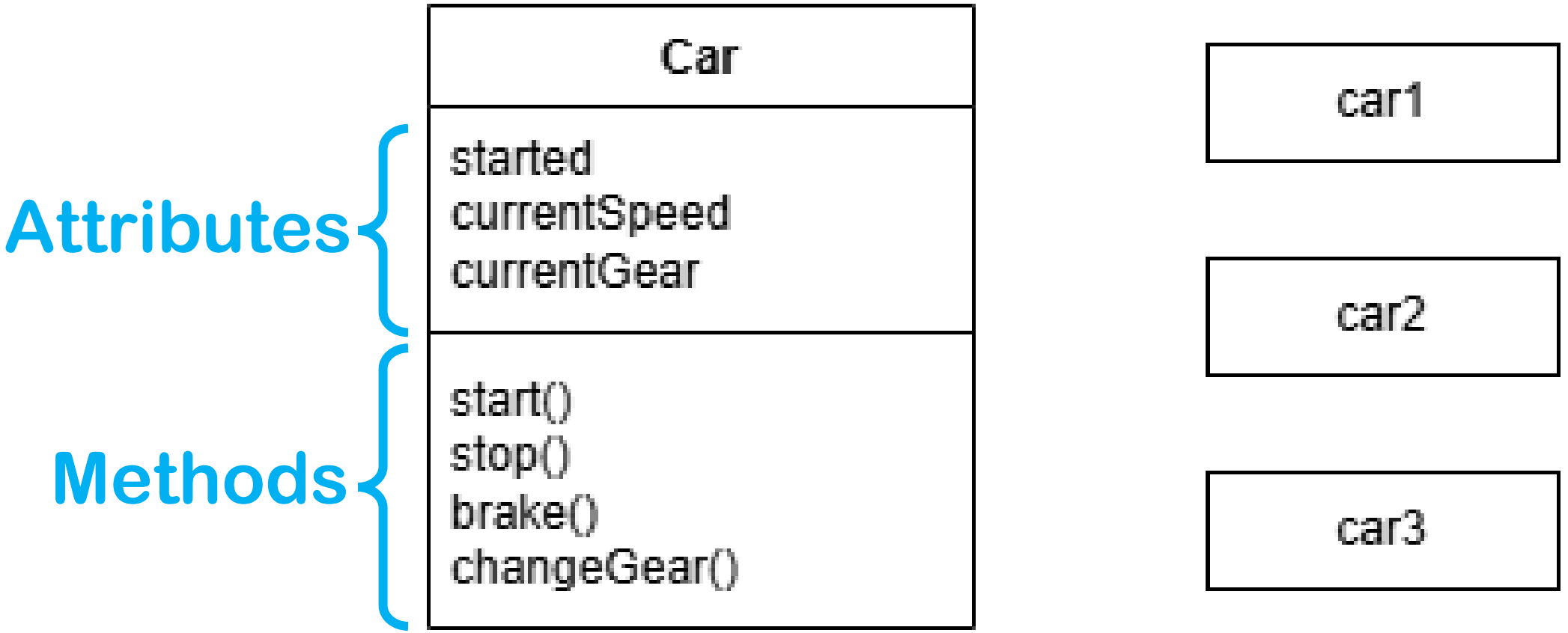



An instance of a class

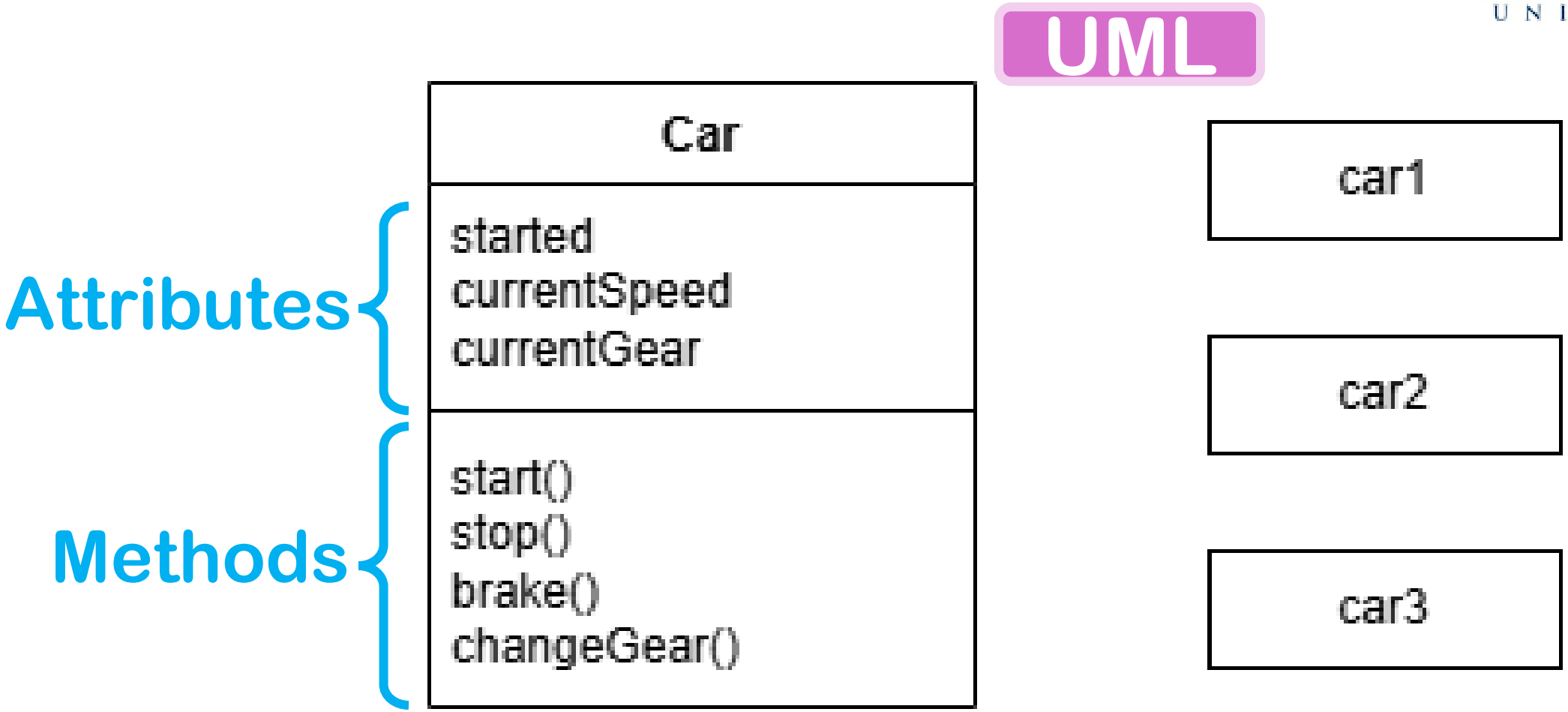
# Class and Object



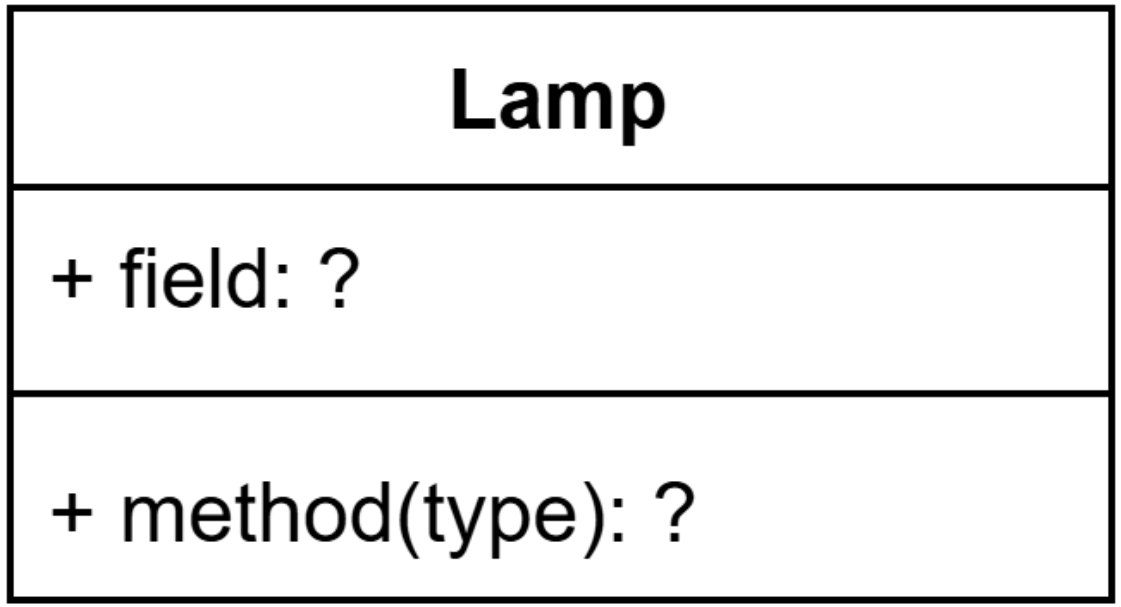
# Class and Object



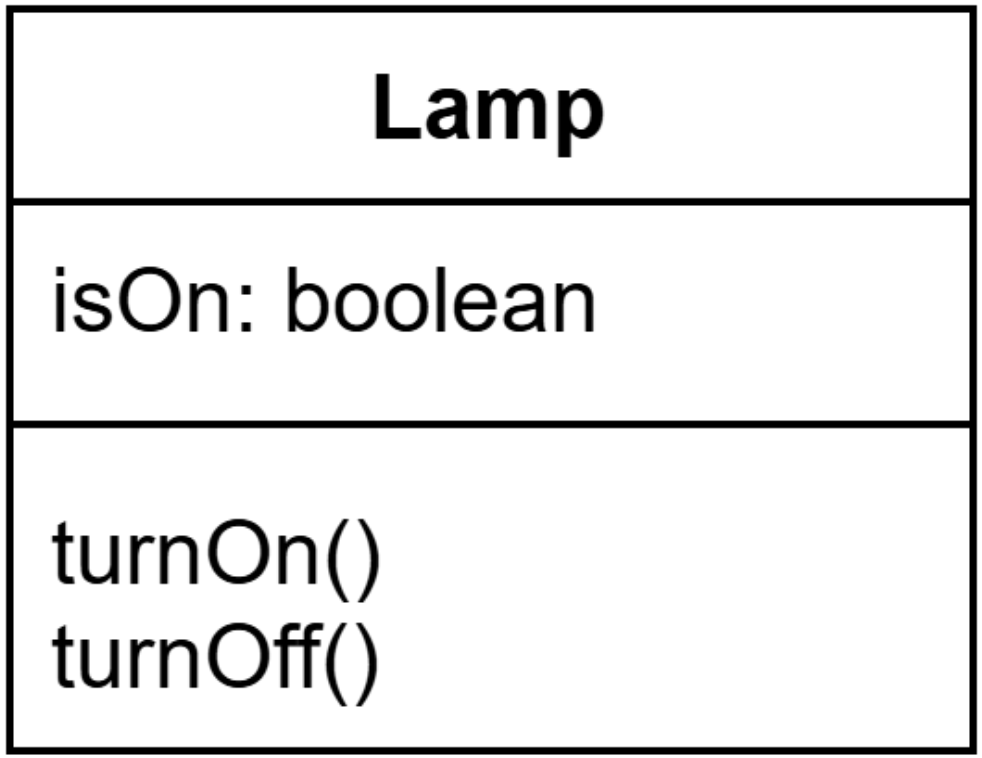
# Class and Object



# Class and Object Example



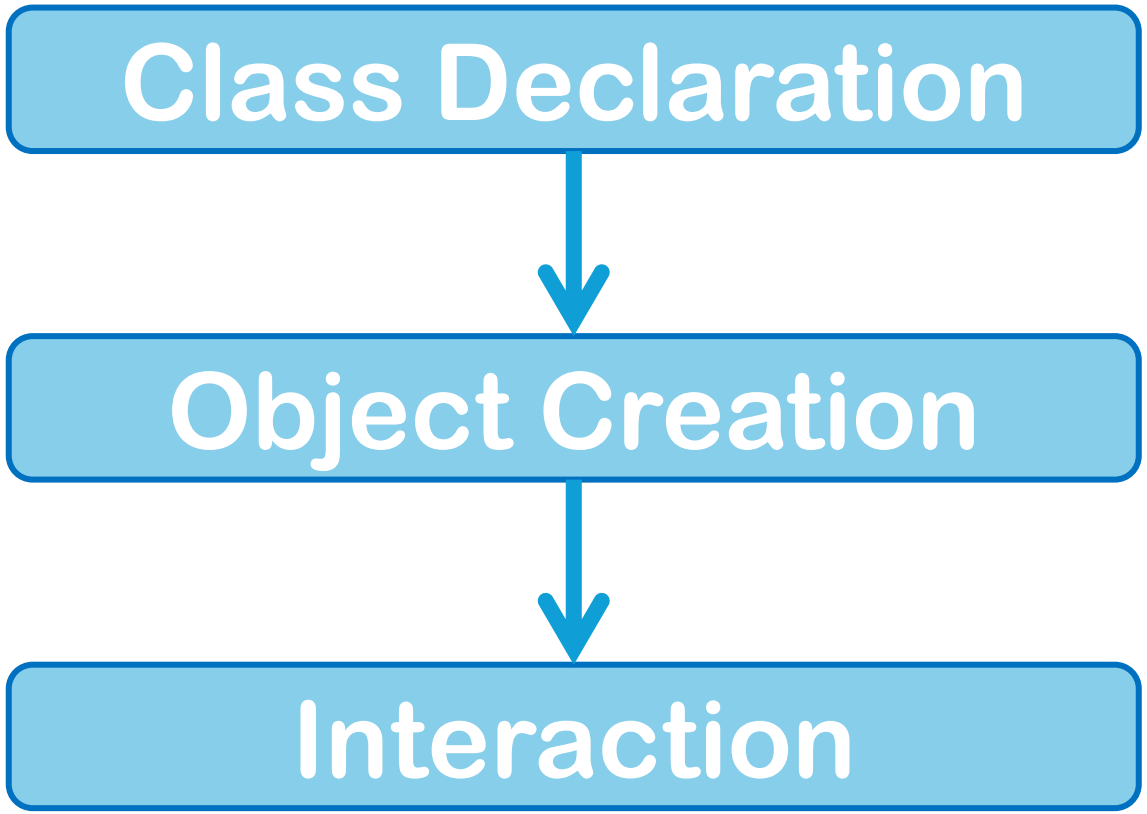
# Class and Object Example



# Class and Object

- Classes: A class is a blueprint or template for creating objects. **It defines the structure and behavior that objects of that class will have**. A class itself does not consume memory until objects are created.
- Objects: An object is an instance of a class. It represents a real-world entity **with specific values for its attributes and can perform actions defined by its class's methods.**

# How they work together ?



# Create a Class and Object

- To create a class, use the keyword class:

src > J Main.java > ...

```
1   public class Main {
2       int x = 5;
3   }
```

- To create an object of Main, specify the class name, followed by the object name, and use the keyword new:

src > J Main.java > ...

```
1   public class Main {
2       int x = 5;
3
        Run | Debug
4       public static void main(String[] args) {
5           Main myObj = new Main();
6           System.out.println(myObj.x);
7       }
8   }
```

# Any Questions?