

A decorative graphic in the top-left corner consisting of three overlapping squares: a light gray one, a medium gray one, and a blue one.

---

# Lecture 5: Java Classes

---

# CONTENTS OF THIS LESSON

- Classes/Objects
  - Class Attributes
  - Constructors
  - User Input
  - Date and Time
  - ArrayList
-



# Classes/Objects



- 
- Java is an object-oriented programming language.
  - Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as driving and braking.
  - A Class is like an object constructor, or a “blueprint” for creating objects.
  - In Java, an object is created from a class. We can first create the class and then create objects through the class.

Create a class called “Main” with a variable x:

```
public class Main {  
    int x = 5;  
}
```

Create an object called “myObj” and print the value of x:

```
public class Main {  
    int x = 5;  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println(myObj.x);  
    }  
}
```

---



# Class Attributes



- 
- We can access attributes by creating an object of the class, and by using the dot syntax ( . )
  - Attribute values can also be modified.

Eg. Create an object of the **Main** class, with the name **myObj** and use the **x** attribute on the object to print its value.

```
public class Main {  
    int x = 5;  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println(myObj.x);  
    }  
}
```

Eg. Change the value of x to 25

```
public class Main {  
    int x = 10;  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        myObj.x = 25; // x is now 25  
        System.out.println(myObj.x);  
    }  
}
```

---



# Constructors



- 
- A constructor in Java is a special method that is used to initialize objects.
  - The constructor is called when an object of a class is created.
  - The constructor can be used to set initial values for object attributes.

```
// Create a Main class
public class Main {
    int x; // Create a class attribute
    // Create a class constructor for the Main class
    public Main() {
        x = 5; // Set the initial value for the class attribute x
    }
    public static void main(String[] args) {
        Main myObj = new Main(); // Create an object of class Main (This will call the constructor)
        System.out.println(myObj.x); // Print the value of x
    }
}
```

**Note:** The constructor name must match the class name, and it cannot have a return type, such as **void**.

---



---

Constructors can also take parameters, which is used to initialize attributes.

```
public class Main {  
    int x;  
    public Main(int y) {  
        x = y;  
    }  
    public static void main(String[] args) {  
        Main myObj = new Main(5);  
        System.out.println(myObj.x);  
    }  
}  
// Outputs 5
```

```
public class Main {  
    int modelYear;  
    String modelName;  
    public Main(int year, String name) {  
        modelYear = year;  
        modelName = name;  
    }  
    public static void main(String[] args) {  
        Main myCar = new Main(1969, "Mustang");  
        System.out.println(myCar.modelYear + " " + myCar.modelName);  
    }  
}  
// Outputs 1969 Mustang
```



# User Input



- 
- In `java.util` package, `Scanner` can be used to get user input.
  - `nextLine()` method is used to read the user input.

### Types of input

Method	Description
<code>nextBoolean()</code>	Reads a boolean value from the user
<code>nextByte()</code>	Reads a byte value from the user
<code>nextDouble()</code>	Reads a double value from the user
<code>nextFloat()</code>	Reads a float value from the user
<code>nextInt()</code>	Reads a int value from the user
<code>nextLine()</code>	Reads a String value from the user

```
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);
        System.out.println("Enter name, age and salary:");
        // String input
        String name = myObj.nextLine();
        // Numerical input
        int age = myObj.nextInt();
        double salary = myObj.nextDouble();
        // Output input by user
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);
    }
}
```



# Date and Time



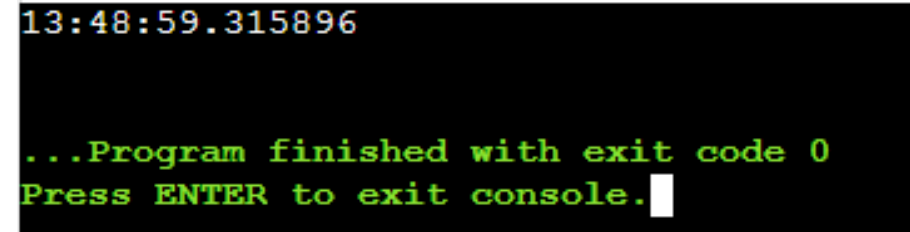
---

Java.time package can be used to work with the date and time API.

Class	Description
LocalDate	Represents a date (year, month, day (yyyy-MM-dd))
LocalTime	Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns))
LocalDateTime	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns)
DateTimeFormatter	Formatter for displaying and parsing date-time objects

- 
- Display the current time.

```
import java.time.LocalDateTime; // import the LocalDateTime class
public class Main {
    public static void main(String[] args) {
        LocalDateTime myObj = LocalDateTime.now(); // Create a date object
        System.out.println(myObj); // Display the current date
    }
}
```

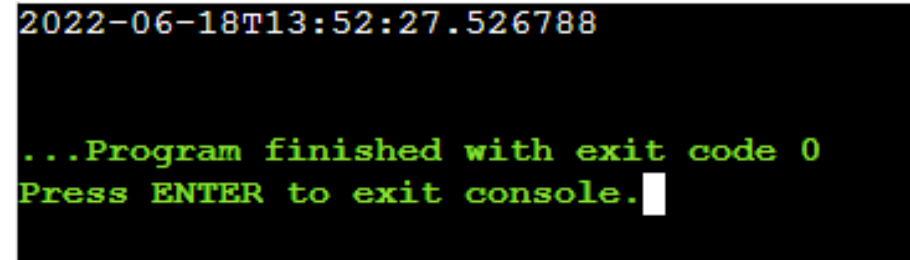


```
13:48:59.315896
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

- Display current date and time

```
import java.time.LocalDateTime; // import the LocalDateTime class
public class Main {
    public static void main(String[] args) {
        LocalDateTime myObj = LocalDateTime.now();
        System.out.println(myObj);
    }
}
```



```
2022-06-18T13:52:27.526788
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

**How can we remove “T” and nanoseconds from the date-time?**

---

---

We can use `DateTimeFormatter` class with `ofPattern()` method to format date-time objects.

```
import java.time.LocalDateTime; // Import the LocalDateTime class
import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter class
public class Main {
    public static void main(String[] args) {
        LocalDateTime myDateObj = LocalDateTime.now();
        System.out.println("Before formatting: " + myDateObj);
        DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        DateTimeFormatter myFormatObj_2 = DateTimeFormatter.ofPattern("E, MMM dd yyyy");
        String formattedDate = myDateObj.format(myFormatObj);
        String formattedDate_2 = myDateObj.format(myFormatObj_2);
        System.out.println("first formatting: " + formattedDate);
        System.out.println("second formatting: " + formattedDate_2);
    }
}
```

```
Before formatting: 2022-06-18T20:20:35.312452
first formatting: 18-06-2022 20:20:35
second formatting: Sat, Jun 18 2022
```

---



# ArrayList





- 
- ArrayList can store strings. It can be found in `java.util` package.
  - `add( )` method can be used to add elements to the ArrayList.
  - `get( )` method can be used to access the element with the index number.
  - `set( )` method can be used to modify the element based on the index number.
  - `remove( )` method can be used to remove an element at certain index.
  - `clear( )` method can be used to delete all elements
  - `size( )` method can be used to find out how many elements in an ArrayList.

```
import java.util.ArrayList;
public class Main
{
    public static void main(String[] args) {
        ArrayList<String> cars = new ArrayList<String>();
        cars.add("Volvo");
        cars.add("BMW");
        cars.add("Ford");
        cars.add("Mazda");
        System.out.println(cars);
        // Access an item
        System.out.println(cars.get(0));
        // Change item
        cars.set(0, "Opel");
        System.out.println(cars);
        // Remove item
        cars.remove(0);
        System.out.println(cars);
        // Size count
        System.out.println(cars.size());
    }
}
```

```
[Volvo, BMW, Ford, Mazda]
Volvo
[Opel, BMW, Ford, Mazda]
[BMW, Ford, Mazda]
3
```

---



Thank you!  
Any questions?

