# LETS LEARN JAVA

## DATA TYPES – BEYOND BASIC TYPES

# Training Plan - Basics

**Topics to be covered**

String – String data type, how its an array of characters, methods on Strings
Immutability of String
Alternatives to String – StringBuilder (and StringBuffer)

Dates - Java 8 time APIs.
LocalDate, LocalDateTime. How to get current date, time. How to operate on dates
Duration, Period and ChronoUnits

Objects – User Defined Types (Classes)
Create instances of user defined Types
Create object instances using Constructors

# Special reference Types - String

Strings are internally array of characters

```
String message = "Hello";
char[] msgArray = {'H', 'e', 'l', 'l', ‘o'};
```

Operations on Strings

**charAt** – find the character at a specific index location
**toCharArray** – converts the string into a character array
**indexOf** – find the position of the first occurrence of a character/string
**contains** – checks if a string is contained within
**substring** – standard substring functionality

# Special reference Types - String

Strings are internally array of characters

```
String message = “  We are learning Java  ";
```

Operations on Strings

**trim** – remove spaces at beginning and end of string
**concat** – concatenate two strings
**toUpperCase** – converts string to UPPER case
**toLowerCase** – converts string to lower case

# Strings are immutable

```
String message = "Learning";
message.concat(" Java");

System.out.println(message);
```

→ "Learning"

What happened here…
Why the concat did not change the value of `message`

# String Alternative – StringBuilder

```java
StringBuilder message = new StringBuilder("Learning");
message.append(" Java");

System.out.println(message);
```

→ "Learning Java"

Operations on `StringBuider` (additional from String)

**reverse** – reverses the string
**setCharAt** – updates the character at a position to the one passed
**insert** – insert characters/string at a position
**delete** – delete characters/string at a position, or between one position to another

There is another alternative called `StringBuffer` which we will discuss post multithreading

# Special reference Types - LocalDate

```
public static void main(String[] args) {
    LocalDate today = LocalDate.now();
    System.out.println(today);

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MMM-yyyy");
    System.out.println(today.format(formatter));
}
```

2021-03-21

21-Mar-2021

LocalDate – for dates
LocalDateTime – for Date and Time

# Date with Time

```
public static void main(String[] args) {
    LocalDate today = LocalDate.now();
    System.out.println(today);

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MMM-yyyy");
    System.out.println(today.format(formatter));

    LocalDateTime now = LocalDateTime.now();
    System.out.println(now);

    DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern("dd-MMM-yyyy hh:mm:ss a");
    System.out.println(now.format(timeFormatter));
}
```

→ 2021-03-27T18:23:24.923820500

→ 27-Mar-2021 06:23:24 PM

# Operations on Dates

```java
public static void main(String[] args) {

    // Set to specific date
    LocalDate someDay = LocalDate.of(1999, 12, 31);
    System.out.println(someDay);

    LocalDate someOtherDay = LocalDate.parse("01-Dec-2020", DateTimeFormatter.ofPattern("dd-MMM-yyyy"));
    System.out.println(someOtherDay);

    LocalDate today = LocalDate.now();

    System.out.println("Today : " + today);
    System.out.println("20 days back : " + today.minusDays(20));
    System.out.println("2 years 3 months from now : " + today.plusYears(2).plusMonths(3));
}
```

# Difference between dates

```
public static void main(String[] args) {

    LocalDate startDate = LocalDate.of(1999, 12, 31);
    LocalDate endDate = LocalDate.of(2021, 03, 27);

    Period period = Period.between(startDate, endDate);

    System.out.println(
            period.getYears() + " Years, " + period.getMonths() + " Months, and " + period.getDays() + " Days");

    System.out.println(ChronoUnit.DAYS.between(startDate, endDate));

}
```

Similar to Period, for calculating difference between two `LocalDateTime`, we use `Duration`.

# ZonedDateTime for working with Time zones

```
public static void main(String[] args) {

    LocalDateTime now = LocalDateTime.now();

    ZonedDateTime nowinIndia = now.atZone(ZoneId.of("Asia/Kolkata"));
    System.out.println(nowinIndia);

    ZonedDateTime nowInLondon = nowinIndia.withZoneSameInstant(ZoneId.of("Europe/London"));
    System.out.println(nowInLondon);

}
```

# User Defined Types - Classes

```java
class Person {

    String name;
    int age;
    boolean isMarried;

    void printYearOfBirth() {
        int birthYear = calculateYearOfBirth();
        System.out.println("You were born in " + birthYear);
    }

    int calculateYearOfBirth() {
        int yearOfBirth = 2021 - age;
        return yearOfBirth;
    }
}
```

**What is a class?** – Attributes and behaviors

**Fundamentals** - Which are the variables here (attributes)

**Eclipse Tricks -** How to refactor and rename variable names

How many methods in this class

How many variables

What will happen if I rename birthyear to yearOfBirth

# Creating instances of Classes

```java
public class MyProgram {

    public static void main(String[] args) {
        Person p = new Person();
        p.name = "Peter";
        p.age = 25;
        p.isMarried = false;

        p.printYearOfBirth();
    }
}
```

What is the meaning of the line

```java
Person p = new Person();
```

What is the difference between the two occurrences of  Person in above line.

**Do it yourself** - What should we change in the program so that output says

"Peter was born in 1996"

# Multiple instances of a class

```java
public class MyProgram {

    public static void main(String[] args) {
        Person peter = new Person();
        peter.name = "Peter";
        peter.age = 25;
        peter.isMarried = false;

        Person paul = new Person();
        paul.name = "Paul";
        paul.age = 29;
        paul.isMarried = true;

        peter.printYearOfBirth();
        paul.printYearOfBirth();

        System.out.println("Peter : " + peter);
        System.out.println("Paul : " + paul);
    }
}
```

What is the meaning of this block

What is the meaning of the values that are printed in place of `peter` and `paul` variables

```
Peter was born in 1996
Paul was born in 1992
Peter : com.vlearntech.java.Person@5ccd43c2
Paul : com.vlearntech.java.Person@4aa8f0b4
```

What happens if I say before the 1st print statement

```
paul = peter;
```

# How to organize this code better

```java
public class MyProgram {

    public static void main(String[] args) {
        Person peter = new Person();
        peter.name = "Peter";
        peter.age = 25;
        peter.isMarried = false;

        Person paul = new Person();
        paul.name = "Paul";
        paul.age = 29;
        paul.isMarried = true;

        peter.printYearOfBirth();
        paul.printYearOfBirth();

        System.out.println("Peter : " + peter);
        System.out.println("Paul : " + paul);
    }
}
```

What is the **problem with the current code**

- Every time you construct a new Person instance, you have to write 4 lines of code
- Imagine if the class had 20 variables (attributes)

# Updated wrapper class

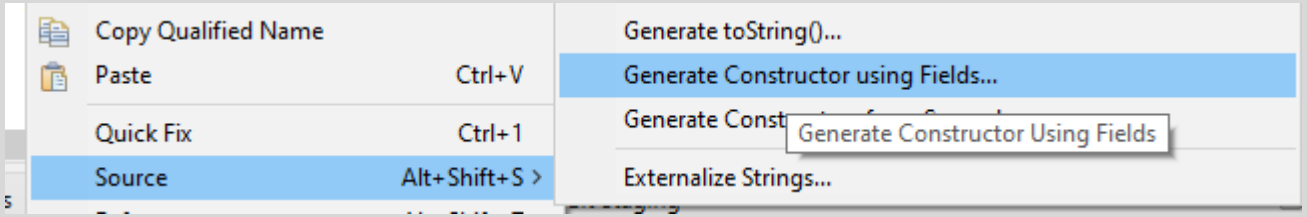
```java
public class MyProgram {

    public static void main(String[] args) {
        Person peter = new Person("Peter", 25, false);
        Person paul = new Person("Paul", 29, true);

        peter.printYearOfBirth();
        paul.printYearOfBirth();

        System.out.println("Peter : " + peter);
        System.out.println("Paul : " + paul);
    }
}
```

Looks neater!

This is called a constructor.

Java provides a default no argument constructor.

You can generate field based constructors in Eclipse by Right click >> Source >> Generate Constructor using Fields



# Lets program the below

- Create a user defined class called Employee with following attributes:
  - Id, Name, Gender, Date of Joining, Salary, Marital Status, Hobbies, Current Project

- Use right data types of each attribute
- Create right constructor using fields
- Create a method to print details of the Employee
- Create 10 instances of Employee and use constructor to populate the details. Create a separate utility class and create a method to generate a list of 10 employee and share that in an array.
- Write a program to loop through all the employees and print details of those whose experience is more than 2 years, and whose salary is less than the average salary of all employees
- Write a program to print details of the employees who have at least 2 hobbies, are not Married and have their names starting with A and have a name of length at least 5.