Day-Three

Notes

Group Coaching

```
1. we share how are our week going and everyone said sunny!(Great)
2. naming the Group Coach (The Four Loops)
```

Read Article

- Auhtor: Sherrie Campbell

  Psychologist, Author, Speaker

  https://www.entrepreneur.com/living/the-7-senses-of-self-development/331147

  - 🌟 **Self-Development**: Success and personal fulfillment come through self-development and exploration. Stay curious and passionate about improving yourself.
  - 🔍 **Self-Awareness**: Begin with a strong sense of self-awareness. Understand your values, beliefs, and larger purpose. Chase your dreams, not others' opinions.
  - 🧠 **Curiosity**: Cultivate an endless curiosity about your future. This inspires resilience and helps you see challenges as opportunities for growth.
  - 🎯 **Direction**: Clarity in self-development provides direction, making decision-making easier and aligning short-term and long-term goals.
  - ✅ **Follow-Through**: Knowing your goals helps you see the benefits of action, even in unenjoyable tasks. Commitment to personal development fuels your willpower.
  - ⏰ **Urgency**: A sense of urgency motivates quick, effective actions. It drives you to assess outcomes efficiently and adapt quickly to changing circumstances.
  - 🔄 **Resiliency**: Develop skills to handle tough times. Resilience allows continuous progress and adaptation, fostering trust in the process.
  - 🤝 **Connectedness**: Improve self-awareness to identify beneficial relationships. Invest in positive connections and cut loose the negative ones.
  - 🚀 **Commitment**: Invest in personal development for lasting success. Continuous learning and self-management are key to achieving greatness.

Warmup Questions

amazed: by how the git checkout command can go back specific point you commit.
No question.

Github and Zip File

How to create a Repository through website?
Creating a new repository on GitHub through the website is a straightforward process. Here's a step-by-step guide:

Step-by-Step Guide to Create a Repository on GitHub:

1. **Log in to GitHub:**
   - Go to GitHub's website and log in to your account.
2. **Go to Your Profile or Organization:**
   - Once logged in, you can either create a repository for your personal account or an organization you're part of. To create it for yourself, click on your profile icon in the upper-right corner and select "Your repositories." For organizations, navigate to the specific organization's page.
3. **Create a New Repository:**
   - In the "Repositories" tab, click on the green **"New"** button or directly go to New Repository.
4. **Set Repository Details:**
   - **Repository Name:** Enter a unique name for your repository.
   - **Description (Optional):** Add a short description of what your repository is about.
   - **Public or Private:** Choose whether the repository will be public (anyone can see it) or private (only you and people you invite can see it).
   - **Initialize Repository:** You have a few options here:
     - **Initialize with README:** Check this box if you want to create a README file right away. This file is useful for providing a description of your project.
     - **.gitignore:** Choose a `.gitignore` template based on the language or framework you plan to use. This will tell Git which files or directories to ignore.
     - **License:** You can also select a license for your project, such as MIT, GPL, or others.
5. **Click "Create Repository":**
   - Once you've filled in all the details, click the green **"Create repository"** button at the bottom.

6. **Repository is Ready:**
    - After creating the repository, you'll be redirected to the main page of the newly created repository, where you can begin uploading files, cloning the repo locally, or collaborating with others.

How to clone the Repository

```
git clone https://github.com/yourname/yearup-first-repo.git
```

What is the Zip file?

A **ZIP file** is a compressed archive format that bundles multiple files or folders into a single file, reducing their overall size using the **DEFLATE** algorithm. It exists to make file storage and transfer more efficient, by reducing file sizes and consolidating multiple items into one.

1. Why ZIP Files Exist:
    - **Save Storage Space**: Compresses files to reduce disk usage.
    - **Faster File Transfer**: Smaller file sizes lead to quicker uploads/downloads.
    - **Convenience**: Combines multiple files/folders into one package.
2. When to Use ZIP Files:
    - **Email Attachments**: To send multiple files in a single attachment.
    - **File Backup**: Compress files to save space and organize backups.
    - **Data Sharing**: Distribute large sets of files more easily by reducing their size.

How does Zip File works?
list of popular compression algorithms and corresponding tools used for ZIP or similar archive formats:

Popular Compression Algorithms (for ZIP or Similar Files):

1. **DEFLATE**:
    - **Used In**: ZIP, GZIP
    - **Description**: A combination of LZ77 (lossless data compression) and Huffman coding. It's the default algorithm for ZIP files.
    - **Tools**: WinRAR, 7-Zip, built-in ZIP support on most operating systems.
2. **LZMA (Lempel-Ziv-Markov Chain Algorithm)**:

- **Used In**: 7z, XZ
- **Description**: High compression ratio but slower speed. Often used in software packaging.
- **Tools**: 7-Zip, XZ Utils.

3. **LZMA2**:
   - **Used In**: 7z, XZ
   - **Description**: An improved version of LZMA, with better support for multi-threading and large files.
   - **Tools**: 7-Zip, XZ Utils.
4. **BZIP2**:
   - **Used In**: .bz2, .tar.bz2
   - **Description**: Provides better compression than DEFLATE but is slower. Uses Burrows-Wheeler Transform (BWT) and Huffman coding.
   - **Tools**: bzip2, tar.
5. **Zstandard (ZSTD)**:
   - **Used In**: .zst
   - **Description**: Modern compression algorithm that balances high compression ratio and fast speed.
   - **Tools**: Zstandard (zstd).
6. **PPMd (Prediction by Partial Matching)**:
   - **Used In**: RAR
   - **Description**: Used for high compression, especially for text data.
   - **Tools**: WinRAR.
7. **LZ77/LZ78 (Lempel-Ziv)**:
   - **Used In**: ZIP (part of DEFLATE), LZH
   - **Description**: Fundamental lossless data compression algorithm, used as the basis for DEFLATE and other algorithms.
   - **Tools**: Various ZIP and LZH utilities.
8. **LZO (Lempel-Ziv-Oberhumer)**:
   - **Used In**: LZO format
   - **Description**: A fast compression algorithm used in real-time data compression.
   - **Tools**: LZOP.

Popular ZIP/Unzip Tools:

1. **7-Zip**:
   - **Supports**: ZIP, 7z, TAR, GZIP, BZIP2, XZ, etc.

- **Description**: Free and open-source, supports a wide range of compression formats with high compression ratio.

2. **WinRAR**:
   - **Supports**: ZIP, RAR, 7z, and more.
   - **Description**: Proprietary tool, famous for handling .rar files and multi-part archives.
3. **WinZip**:
   - **Supports**: ZIP, ZIPX, RAR, 7z, GZIP, etc.
   - **Description**: One of the original ZIP file tools, with added features for encryption and cloud integration.
4. **Gzip**:
   - **Supports**: GZIP (.gz)
   - **Description**: Common on Unix/Linux systems for compressing single files, often used with `tar`.
5. **bzip2**:
   - **Supports**: BZIP2 (.bz2)
   - **Description**: Used in Unix/Linux for high-compression archives, often combined with `tar`.
6. **Tar**:
   - **Supports**: TAR (.tar), TAR.GZ, TAR.BZ2, TAR.XZ
   - **Description**: Unix/Linux archiving tool used with various compression algorithms like gzip, bzip2, and xz.
7. **XZ Utils**:
   - **Supports**: XZ, LZMA
   - **Description**: Toolset for handling `.xz` and `.lzma` files, providing high compression with LZMA algorithms.
8. **Zstandard (zstd)**:
   - **Supports**: ZST (.zst)
   - **Description**: Compression tool optimized for speed and a good compression ratio.

How to use IntelliJ IDE

Introduction to IntelliJ IDEA

IntelliJ IDEA is a popular integrated development environment (IDE) developed by JetBrains, primarily used for Java development, but it also supports a wide variety of other programming languages such as Kotlin, Scala, Groovy, Python, JavaScript, and more. It's known for its advanced coding assistance, developer-friendly features, and wide plugin ecosystem. Here's an overview of IntelliJ IDEA:

Key Features of IntelliJ IDEA:

1. **Smart Code Completion**: IntelliJ IDEA provides context-aware code completion, suggesting the most relevant options based on the code you're writing.
2. **Powerful Refactoring Tools**: It offers various refactoring options, such as renaming, moving, extracting methods, and changing method signatures, ensuring code quality and maintainability.
3. **Built-in Version Control**: IntelliJ integrates with popular version control systems like Git, GitHub, Mercurial, and Subversion, allowing seamless project management.
4. **Code Navigation**: IntelliJ allows easy navigation through code, with features like "Go to Declaration," "Find Usages," and jumping to the implementation of interfaces, making it easy to work on large projects.
5. **Debugging and Testing**: It has robust debugging tools, including breakpoints, watches, and variable inspection. For testing, IntelliJ integrates with JUnit, TestNG, and other popular testing frameworks, allowing you to write and run tests directly within the IDE.
6. **Plugin Support**: IntelliJ IDEA has a vast ecosystem of plugins that extend its functionality, from language support to integration with cloud services and tools like Docker, Kubernetes, and databases.
7. **Project Templates and Framework Support**: It supports various project types and frameworks such as Spring Boot, Maven, Gradle, and Java EE, simplifying setup and configuration.
8. **Integrated Tools**: The IDE comes with tools for database management, terminal access, and deployment, allowing you to handle many aspects of development without leaving the IDE.

Why Use IntelliJ IDEA?

- **Productivity**: IntelliJ IDEA is designed to boost developer productivity with features like on-the-fly code analysis, advanced code generation, and intelligent suggestions.
- **Cross-Language Refactoring**: You can refactor not just within a language, but across different languages used in your project.
- **Customization**: From themes to keymaps and plugins, IntelliJ offers a high degree of customization, letting you configure the IDE to suit your workflow.

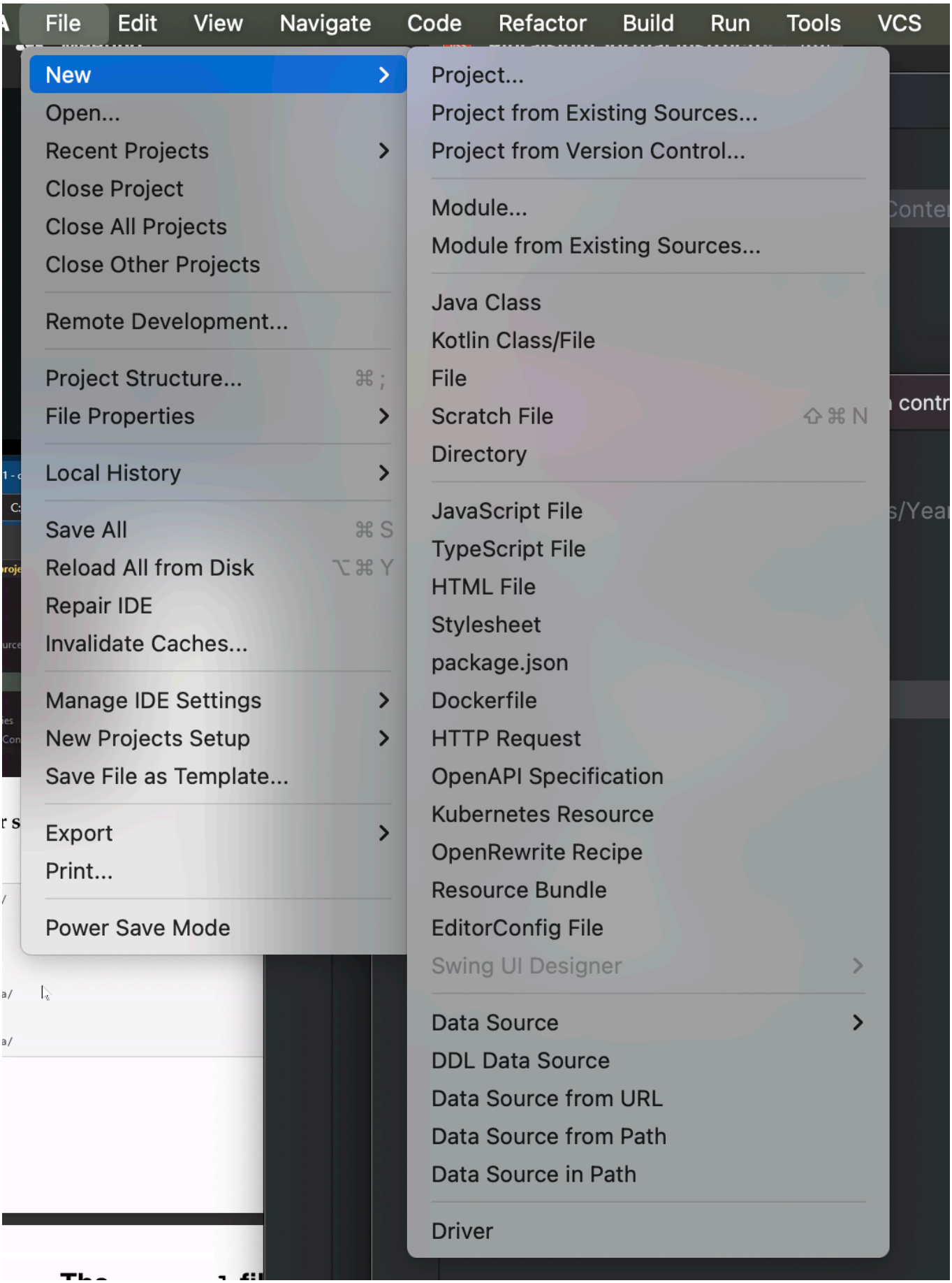
Editions:

- **IntelliJ IDEA Community Edition**: Free and open-source, this version supports JVM-based and Android development.

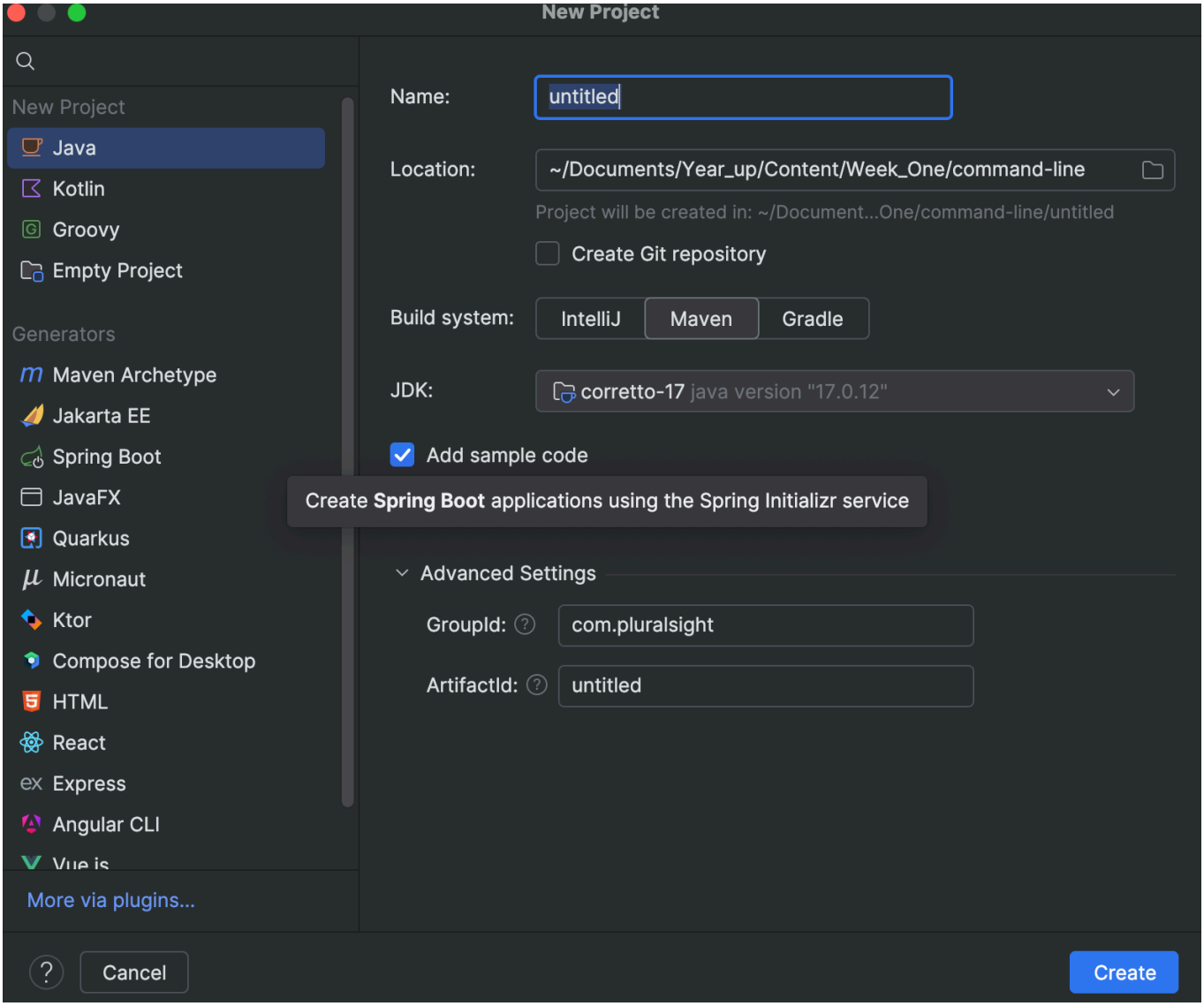
- **IntelliJ IDEA Ultimate Edition**: Paid version with additional features like support for enterprise frameworks, advanced database tools, and web development features.

For Java development, IntelliJ IDEA is widely regarded as one of the best IDEs due to its extensive feature set and user-friendly design.

Step of Creating a New Java Maven Project

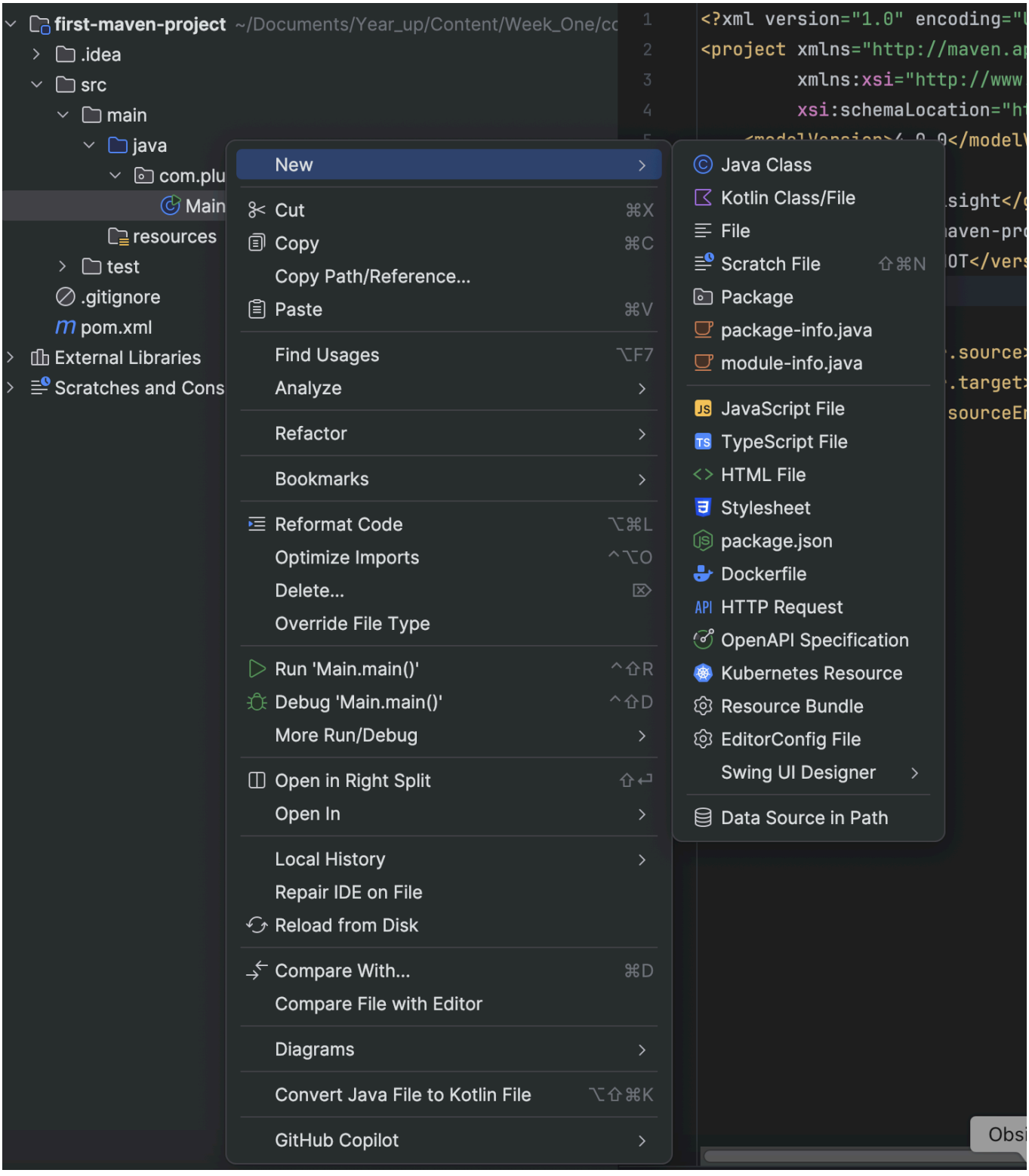

File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
New
Open...
Recent Projects
Close Project
Close All Projects
Close Other Projects
Remote Development...
Project Structure... ⌘ ;
File Properties
Local History
Save All ⌘ S
Reload All from Disk ⌥ ⌘ Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Project...
Project from Existing Sources...
Project from Version Control...
Module...
Module from Existing Sources...
Java Class
Kotlin Class/File
File
Scratch File ⇧ ⌘ N
Directory
JavaScript File
TypeScript File
HTML File
Stylesheet
package.json
Dockerfile
HTTP Request
OpenAPI Specification
Kubernetes Resource
OpenRewrite Recipe
Resource Bundle
EditorConfig File
Swing UI Designer
Data Source
DDL Data Source
Data Source from URL
Data Source from Path
Data Source in Path
Driver

Click the Project

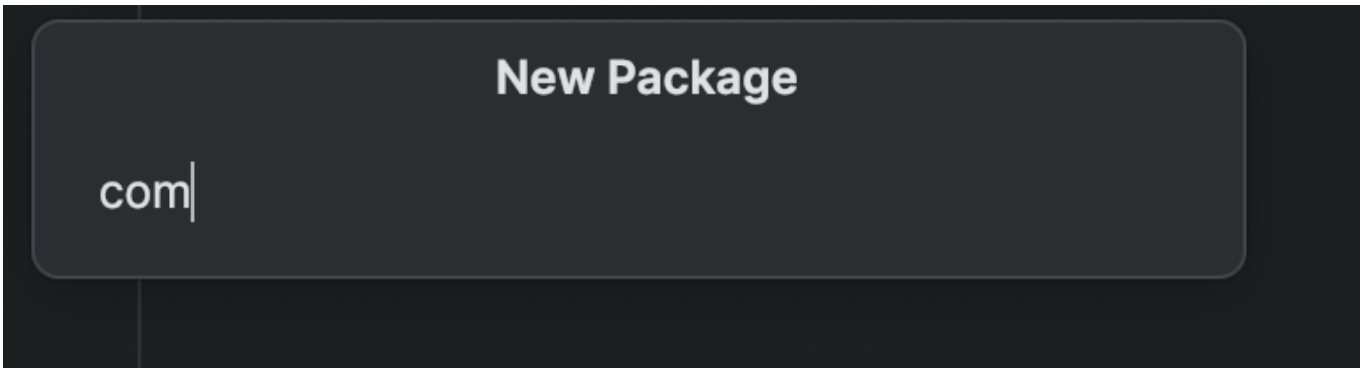


1. On the left side, You will need to click the Java.
2. On the right side,
    1. Name : Your project name
    2. location: Where you want to your project to store
    3. Create Git Repository: git init
    4. Build System: You will pick one of them, Mainly Maven or Gradlle
    5. JDK: Pick the Java version
    6. groupId: create two more level of folders deep. com is a folder, come with . pluralsight also is a folder but inside of the com.
    7. ArtifactId: Name of your project

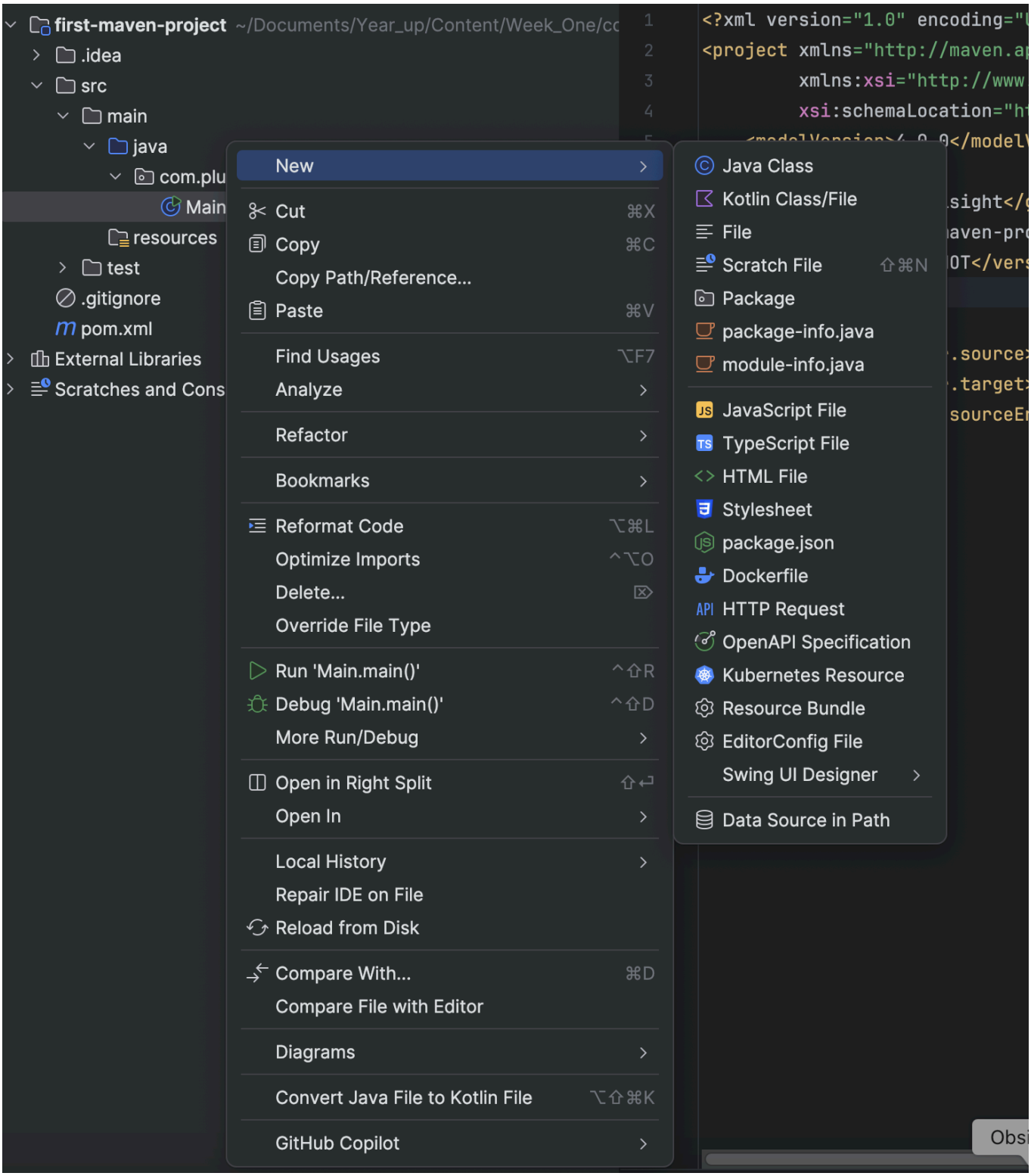
How to Create a Java File or Package



See the above image, you will notice that there is a package option, click the package, you will have a package create.



Type the package name.



See the above image, you will notice that there is a Java Class option, click the Java Class , you will have a Java Class created.

Java Syntax & Git Organization repository & Naming Conventions

1. Main method is the main entry point of your Java Project

2. Like English grammar. Programming languages also have their own rules. Syntax
    1. Creating Organization
        1. Creating Repository inside of the Organization
        2. `git clone https://github.com/your-organization/shopping-list.git`
        3. creating a java project in the repository
        2.

```
1279  git status
1280  git add .
1281  git commit -m "delete the main.java and create ShoppingList.java"
1282  git push
1283  git status
1284  git log
1285  ls
```

EXERCISE 1

Using IntelliJ, create a new Java application that will list at least 10 items that should be on your shopping list.

1. Create a new package named `com.pluralsight`
2. In the `com.pluralsight` package create a new java class named `ShoppingList`. Remember it must be in a .java file of the same name.
3. Within the `ShoppingList` class, create a `main` method.
4. In the `main()` method use the `System.out.println()` to display a shopping list with at least 10 items.
5. Run the program. If there are any errors, fix them and run it again.
6. Push your changes to GitHub (always stage, commit and push your changes)
    i. `git add -A`
    ii. `git commit -m "completed ShoppingList app"`
    iii. `git push origin main`


```
1279  git status
1280  git add .
1281  git commit -m "delete the main.java and create ShoppingList.java"
1282  git push
1283  git status
1284  git log
1285  ls
```


Java Style Guide

Naming Conventions

1. Camel Case (camelCase)

- Format: First word starts with a lowercase letter, and each subsequent word starts with an uppercase letter.
- Example: `myVariableName`, `getUserInfo`
- Common usage: Variable names, function/method names in languages like JavaScript, Java, and C#.

2. Pascal Case (PascalCase)

- Format: Each word starts with an uppercase letter.
- Example: `MyClassName`, `GetUserInfo`
- Common usage: Class names, types, namespaces in languages like C#, Java, and TypeScript.

3. Snake Case (snake_case)

- Format: All letters are lowercase, and words are separated by underscores.
- Example: `my_variable_name`, `get_user_info`
- Common usage: Variable names, function names, and file names in languages like Python, Ruby, and C.

4. Screaming Snake Case (SCREAMING_SNAKE_CASE)

- Format: All letters are uppercase, and words are separated by underscores.
- Example: `MAX_VALUE`, `PI_CONSTANT`
- Common usage: Constants in many languages like Python, Java, C, and C++.

5. Kebab Case (kebab-case)

- Format: All letters are lowercase, and words are separated by hyphens.
- Example: `my-variable-name`, `get-user-info`
- Common usage: URLs, CSS class names, and file names in some web development scenarios.

6. Lowercase (lowercase)

- Format: All letters are lowercase without any separation.

- Example: `myvariablename`, `myfunction`
- Common usage: Short variable names, some database table names, and command-line flags.

7. Uppercase (UPPERCASE)

- Format: All letters are uppercase without any separation.
- Example: `MYVARIABLENAME`
- Common usage: Often used for constants in C or embedded systems.

8. Hungarian Notation

- Format: Prefixes are added to variable names to indicate type or usage.
- Example: `strName`, `iCount`, `arrValues`
- Common usage: Historically used in languages like C/C++ and Visual Basic.

9. Dot Notation

- Format: Words are separated by dots.
- Example: `my.variable.name`
- Common usage: Package names, namespaces, and file paths in languages like Java and Python.

10. Train Case (Train-Case)

- Format: Similar to kebab case, but each word starts with a capital letter.
- Example: `My-Variable-Name`, `Get-User-Info`
- Common usage: Less common, but sometimes seen in certain environments for filenames.

11. MACRO_CASE

- Format: Similar to SCREAMING_SNAKE_CASE but typically used in C macros.
- Example: `#define MAX_BUFFER_SIZE 1024`
- Common usage: Used in C/C++ preprocessor macros.

These conventions help in creating readable, organized, and maintainable code. The choice of convention often depends on the language, framework, and project guidelines.

Java Syntax

1. Basic Program Structure

Every Java program consists of classes, methods, and statements. A simple program starts with a class and includes a `main` method, which is the entry point of the application.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");  // Output: Hello, World!
    }
}
```

2. Keywords

Keywords in Java are reserved words that have special meaning and cannot be used as identifiers (e.g., variable names). Here's a small program using some Java keywords.

```
public class Example {
    public static void main(String[] args) {
        int age = 25; // 'int' is a keyword used to declare an integer
variable
        if (age >= 18) { // 'if' and 'else' are control flow keywords
            System.out.println("You are an adult.");
        } else {
            System.out.println("You are a minor.");
        }
    }
}
```

3. Data Types

Java is a statically-typed language. You need to declare the type of variable before using it.

Primitive Data Types Example:

```java
public class DataTypesExample {
    public static void main(String[] args) {
        int age = 24;            // Integer
        double pi = 3.14159;     // Floating-point number
        char grade = 'A';        // Single character
        boolean isJavaFun = true; // Boolean (true/false)
        /*
        Primitive data types -
        includes
        - byte,
        - short,
        - int,
        - long,
        - float,
        - double,
        - boolean,
        - char,
        */

        System.out.println("Age: " + age); // Age: 24
        System.out.println("PI: " + pi); // PI : 3.14159
        System.out.println("Grade: " + grade); // Grade: A
        System.out.println("Is Java fun? " + isJavaFun); // Is Java fun?
True
    }
}
```

Reference Data Type Example:

```java
public class ReferenceTypesExample {
    public static void main(String[] args) {
        String greeting = "Hello, Java!";  // String is a reference type
        System.out.println(greeting);  // Output: Hello, Java!
    }
}
```

4. Variables

Variables hold values in Java and must be declared with a data type.

```java
public class VariablesExample {
    public static void main(String[] args) {
        int x = 10;
        double y = 5.5;
        String name = "Yiming";

        System.out.println("x: " + x);
        System.out.println("y: " + y);
        System.out.println("Name: " + name);
    }
}
```

5. Operators

Operators are used to perform operations on variables and values.

Arithmetic Operators Example:

```java
public class ArithmeticExample {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        System.out.println("Addition: " + (a + b));    // Output: 15
        System.out.println("Subtraction: " + (a - b)); // Output: 5
        System.out.println("Multiplication: " + (a * b)); // Output: 50
        System.out.println("Division: " + (a / b));   // Output: 2
        System.out.println("Modulus: " + (a % b));    // Output: 0
    }
}
```

Comparison and Logical Operators Example:

```java
public class ComparisonExample {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        System.out.println("a == b: " + (a == b));  // Output: false
        System.out.println("a > b: " + (a > b));    // Output: true
```

```
        System.out.println("a < b: " + (a < b));    // Output: false
        System.out.println("a != b: " + (a != b));  // Output: true
        System.out.println("a > 0 && b > 0: " + (a > 0 && b > 0)); //
Output: true
        System.out.println("a > 0 || b < 0: " + (a > 0 || b < 0)); //
Output: true
    }
}
```

6. Control Flow Statements

`if-else` Statement Example:

```
public class IfElseExample {
    public static void main(String[] args) {
        int score = 85;

        if (score >= 90) {
            System.out.println("Grade: A");
        } else if (score >= 80) {
            System.out.println("Grade: B");
        } else {
            System.out.println("Grade: C");
        }
    }
}
```

`switch` Statement Example:

```
public class SwitchExample {
    public static void main(String[] args) {
        int day = 2;

        switch (day) {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            default:
                System.out.println("Other day");
```

```
                break;
            }
        }
    }
```

Loops:

`for` Loop Example:

```
public class ForLoopExample {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            System.out.println(i);
        }
    }
}
```

`while` Loop Example:

```
public class WhileLoopExample {
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println(i);
            i++;
        }
    }
}
```

`do-while` Loop Example:

```
public class DoWhileExample {
    public static void main(String[] args) {
        int i = 0;
        do {
            System.out.println(i);
            i++;
        } while (i < 5);
```

```
    }
}
```

7. Methods

Methods are reusable blocks of code that perform specific tasks.

```
public class MethodExample {
    public static int add(int a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        int result = add(5, 3);
        System.out.println("Result: " + result);  // Output: 8
    }
}
```

8. Classes and Objects

In Java, classes are blueprints for objects.

```
public class Car {
    String model;
    int year;

    public Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    public void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }

    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2020);
        car1.displayInfo();  // Output: Model: Toyota, Year: 2020
    }
}
```

9. Arrays

Arrays store multiple values of the same type.

```
public class ArrayExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

10. Exception Handling

Java uses `try-catch` blocks to handle exceptions (errors) gracefully.

```
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;  // This will throw an exception
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero.");
        } finally {
            System.out.println("This block always runs.");
        }
    }
}
```

Question / Inspired

1. **Inspired** : I am inspired that we can create our own organization on Github to organize repositories
2. **Question** : How can we add only one file or specific on github instead of add all of them `git add .`?