



# CIT-260

## Week One

## Objectives for this week

At the end of this week, students should be able to

- Explain the contents of the syllabus
- Explain what a high level programming language is
- Describe the steps to build and execute a Java program
- Install and configure IntelliJ IDEA
- Describe the structure of a Java program
- Write, compile, and execute a simple Java program

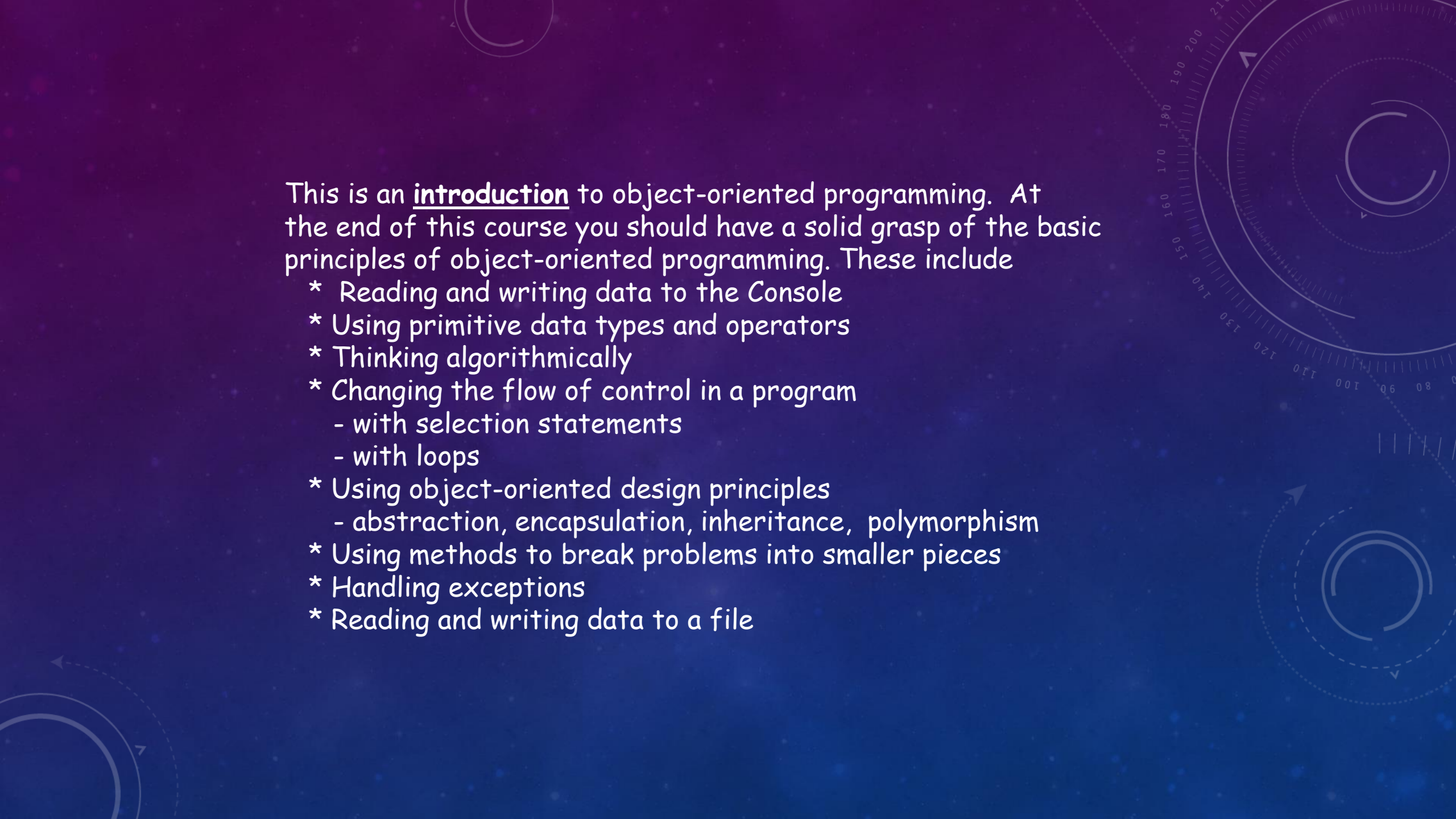
## What You Will Learn This Week

- What this course is all about
- What is in the Course Syllabus
- A little about computer programs
- What a simple Java program looks like
- Who your classmates are and something about them

## What You Will Do This Week

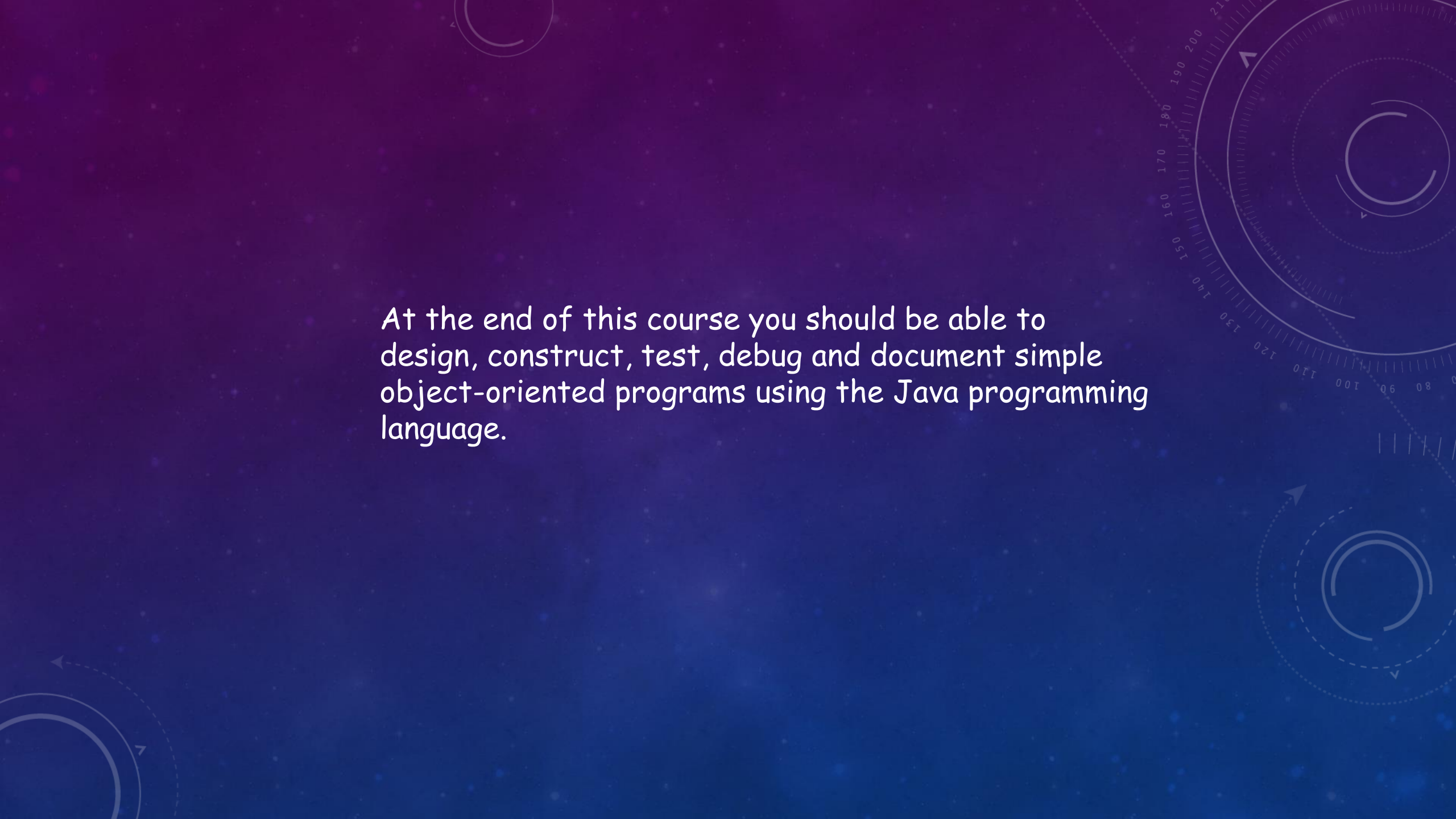
- Study this set of slides
- Review the course syllabus
- Get to know your class-mates
- Sign up for a Zoom account
- Install IntelliJ IDEA
- Write a short Java program and submit it



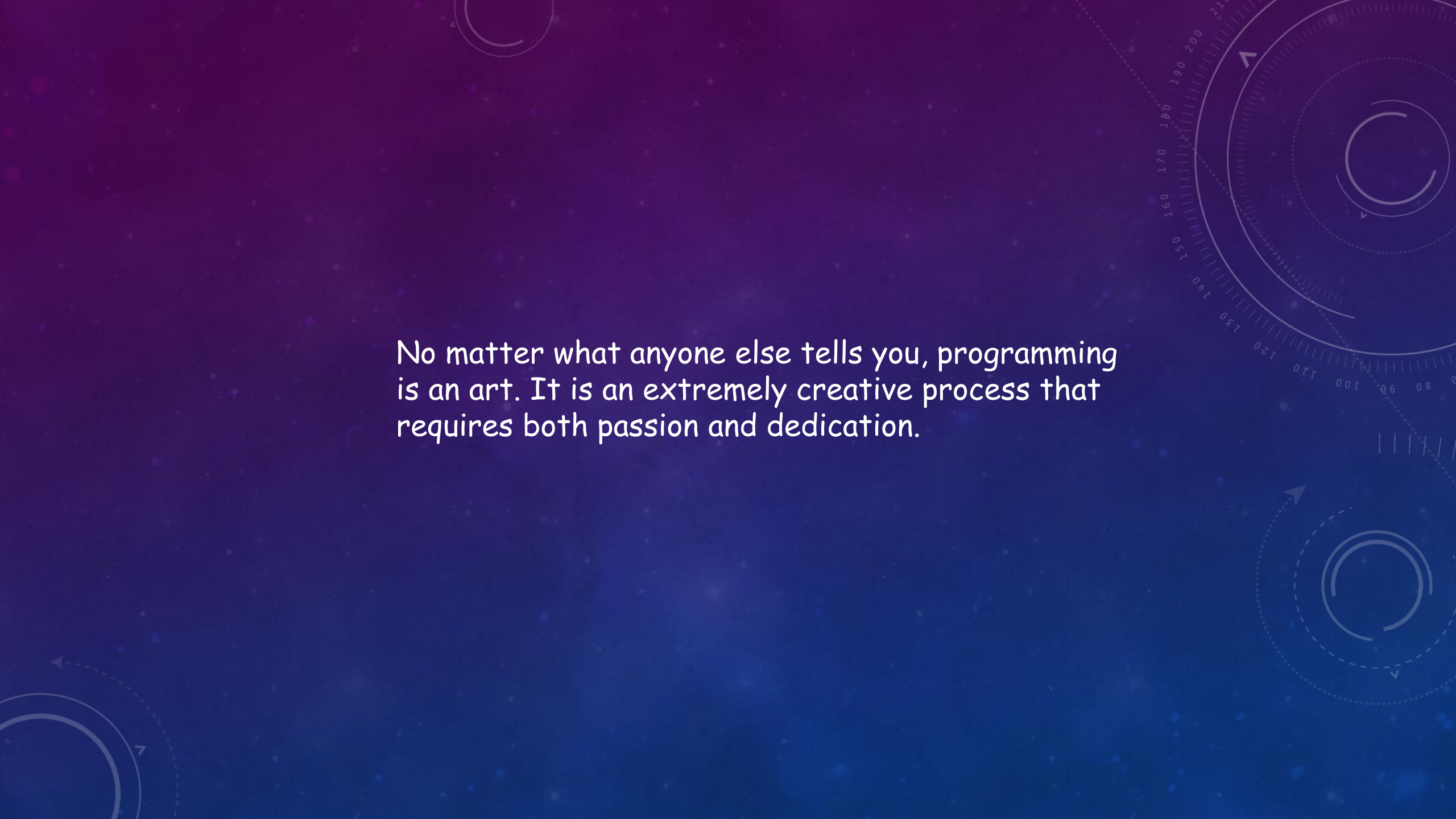


This is an introduction to object-oriented programming. At the end of this course you should have a solid grasp of the basic principles of object-oriented programming. These include

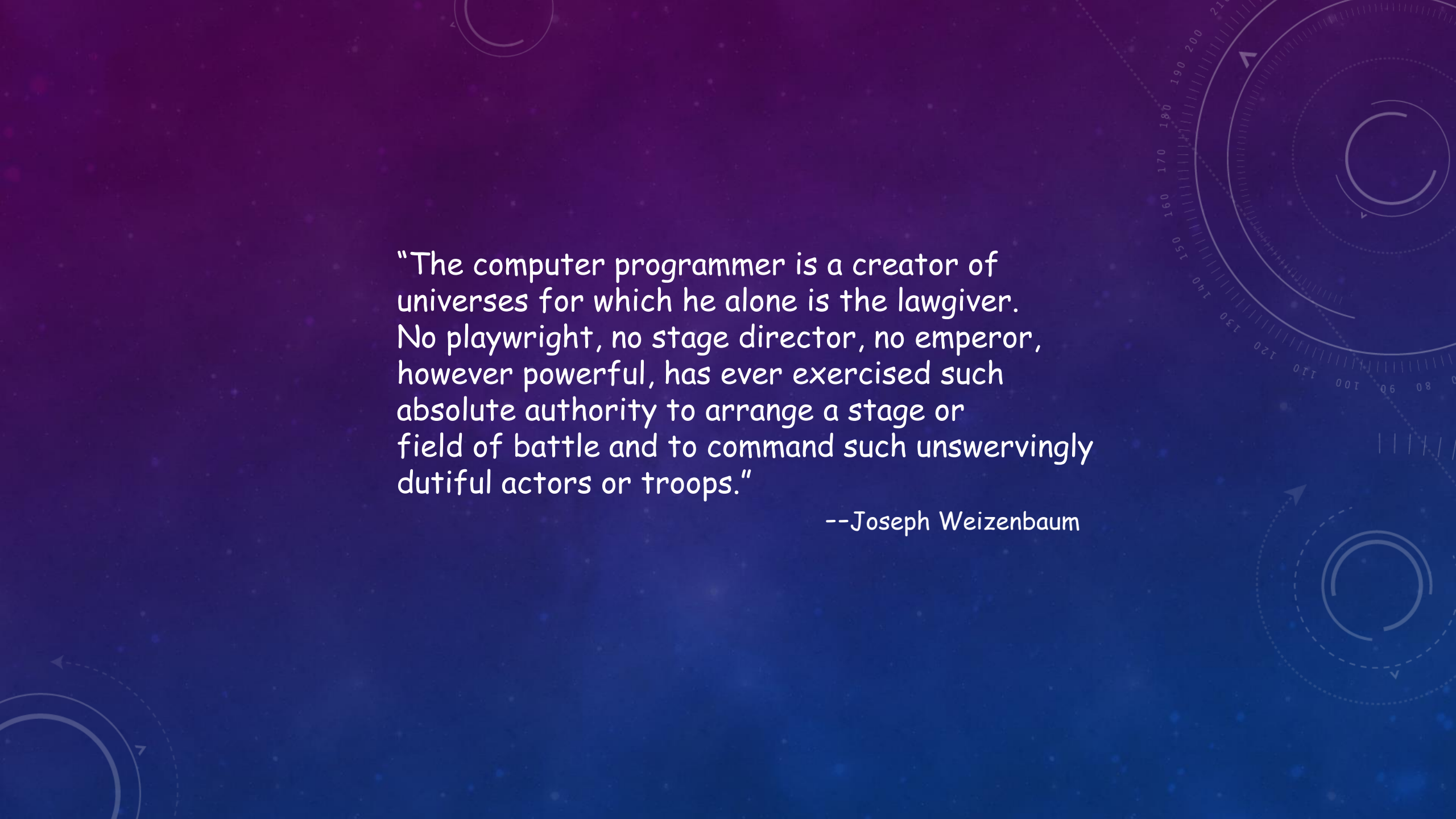
- \* Reading and writing data to the Console
- \* Using primitive data types and operators
- \* Thinking algorithmically
- \* Changing the flow of control in a program
  - with selection statements
  - with loops
- \* Using object-oriented design principles
  - abstraction, encapsulation, inheritance, polymorphism
- \* Using methods to break problems into smaller pieces
- \* Handling exceptions
- \* Reading and writing data to a file

The background is a deep blue gradient with a subtle pattern of white dots, resembling a starry night sky. Overlaid on this are several faint, white geometric shapes: concentric circles and arcs. In the upper right, a large circular scale with degree markings (90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210) is visible. In the lower right, there are dashed circular paths with arrows indicating a clockwise direction. In the lower left, a solid circular path with an arrow indicates a counter-clockwise direction. The text is centered in the middle-left portion of the image.

At the end of this course you should be able to design, construct, test, debug and document simple object-oriented programs using the Java programming language.



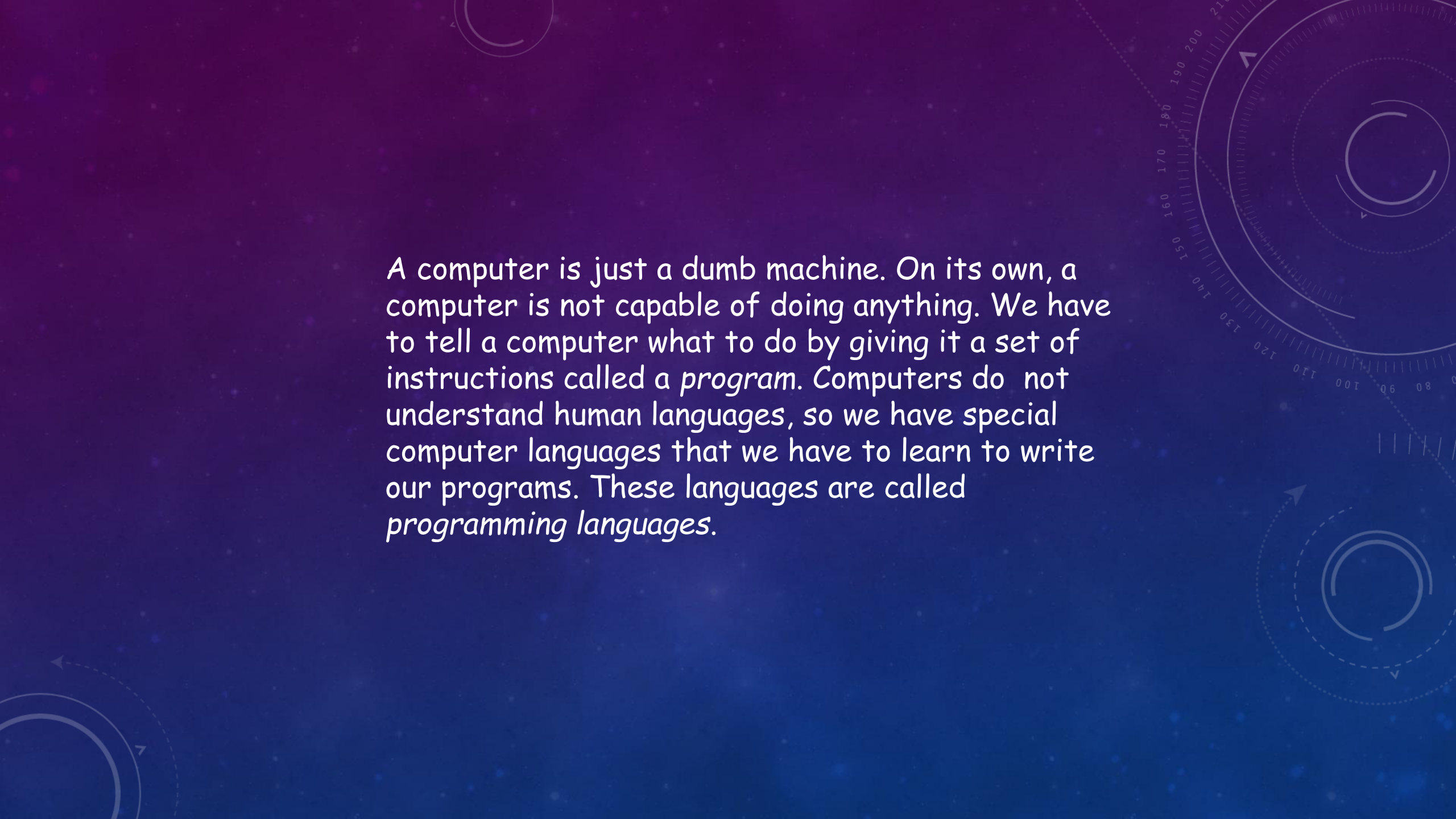
No matter what anyone else tells you, programming is an art. It is an extremely creative process that requires both passion and dedication.



"The computer programmer is a creator of universes for which he alone is the lawgiver. No playwright, no stage director, no emperor, however powerful, has ever exercised such absolute authority to arrange a stage or field of battle and to command such unswervingly dutiful actors or troops."

--Joseph Weizenbaum





A computer is just a dumb machine. On its own, a computer is not capable of doing anything. We have to tell a computer what to do by giving it a set of instructions called a *program*. Computers do not understand human languages, so we have special computer languages that we have to learn to write our programs. These languages are called *programming languages*.

# Machine Language

The most primitive form of programming language is machine language. Every computer has a built in set primitive instructions, which are entered in binary. Entering binary instructions into the computer is a tedious process. The picture here shows a Digital Equipment PDP-8 computer, a popular computer in the early 1970s. Programs could be entered on the front panel by flipping switches. When the switch was up it represented a zero. When it was down it represented a one.



# Assembly Language

Assembly languages were developed to make programming easier. A program called the assembler is used to convert assembly language programs into machine code. This conversion is very fast, since there is usually a one-to-one relationship between assembly language code and machine code. Here is an example of a PDP-8 assembly language program that adds the contents of two memory addresses and leaves the sum in the accumulator

```
00200 CLA           / Clear the accumulator
00201 TAD A         / Add contents of memory location A to the
accumulator
00202 TAD B         / Add the contents of memory location B to the
accumulator
00203 HLT           / Stop the CPU
00204 JMP I, 7600    / return control to the operating system
00205 A, 0003       / define memory location A and store the value
of 3 there
00206 B, 004        / define memory location B and store the value of
4 there
```



# High Level Languages

High-level programming languages are English-like and are much easier to learn and to use than assembly language. For example, the following is a high-level language statement that adds the values of a and b, and stores the sum in c.

$$c = a + b$$



## Procedural Programming

In procedural programming languages, like *C*, statements such as

$c = a + b$

are combined together into units known as procedures, functions, or subroutines. Procedures can be called from anywhere in a program and they operate on the data values (like *a*, *b*, and *c*) in the program. Much of the data in a procedural program is defined in a way that it can be seen and operated on by any procedure in the program. This separation of data from the procedures that work it makes programs that are difficult to maintain, and there is limited re-use of procedures.

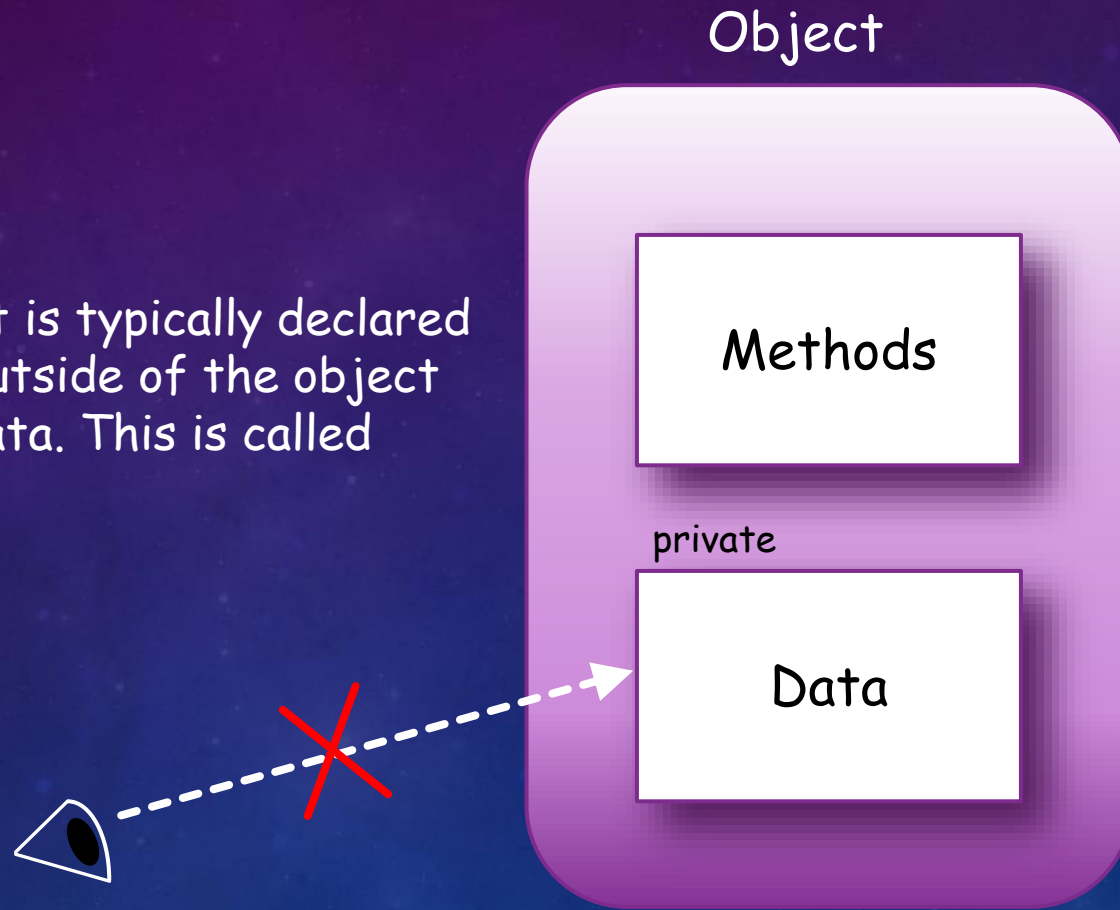
In an object-oriented programming language data and the procedures that operate on that data are put into a software package called an *object*. In Java, we refer to the procedures that are associated with an object as *methods*.

## Object

Methods

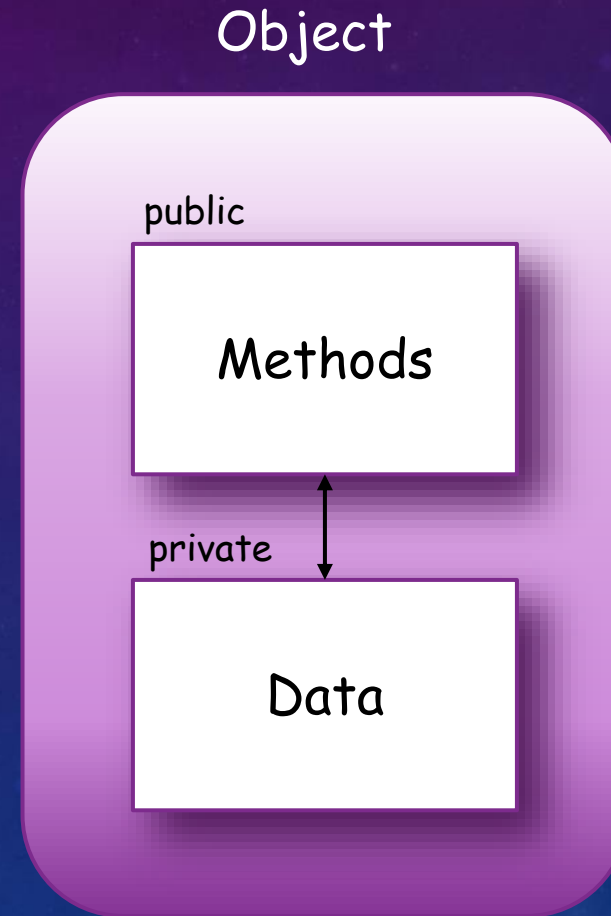
Data

The data inside of the object is typically declared as private, so that entities outside of the object cannot directly access the data. This is called *data hiding*.

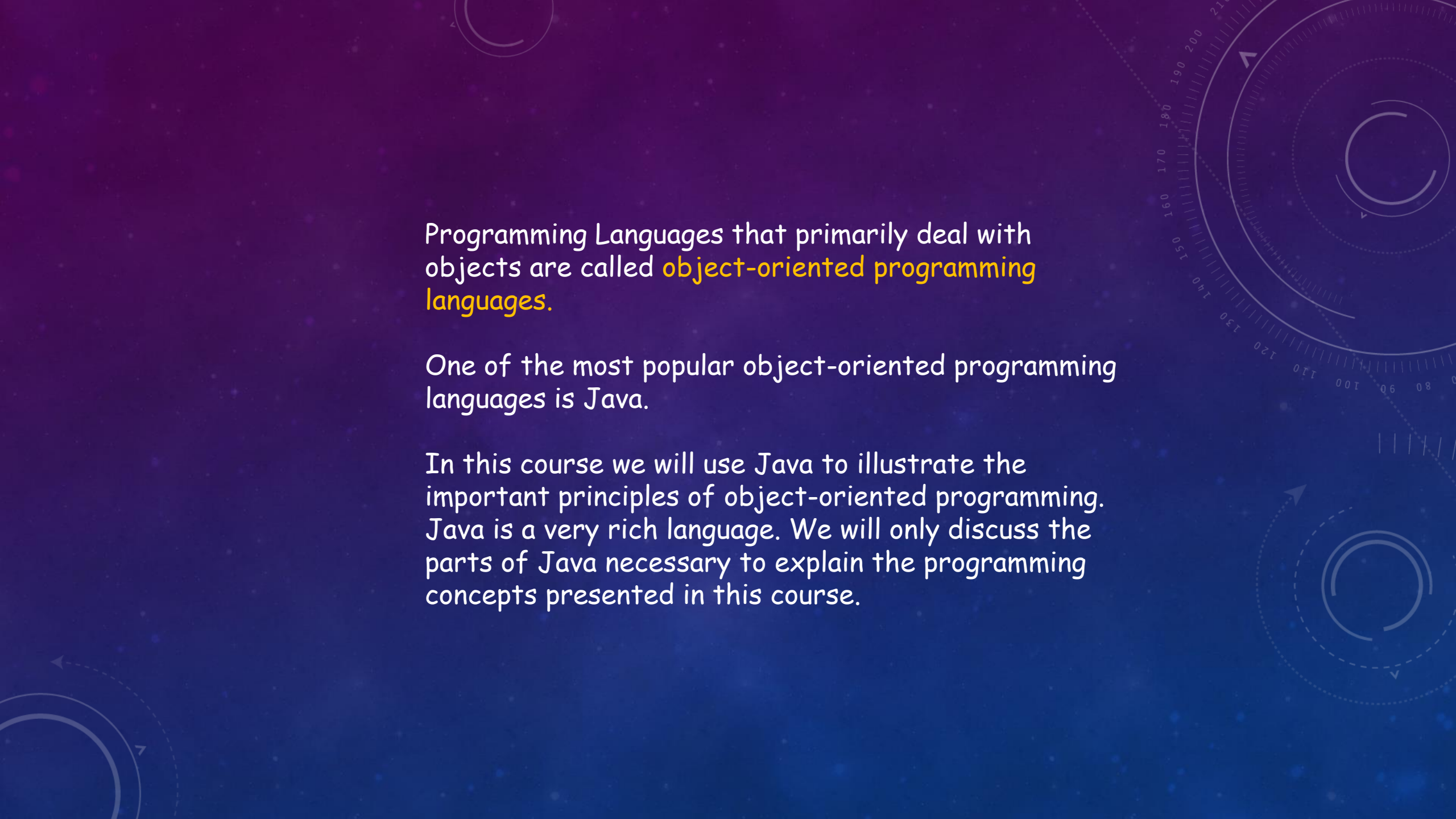


The methods inside of the object are typically declared as public, so that entities outside of the object can use them to access and manipulate the data in the object.

Packaging the data and methods together this way is called *encapsulation*.







Programming Languages that primarily deal with objects are called **object-oriented programming languages**.

One of the most popular object-oriented programming languages is Java.

In this course we will use Java to illustrate the important principles of object-oriented programming. Java is a very rich language. We will only discuss the parts of Java necessary to explain the programming concepts presented in this course.



Java was developed by James Gosling at Sun Microsystems. It was introduced in 1995 as part of Sun's Java platform.

Java derives much of its syntax from *C* and *C++*, but unlike *C* and *C++*, Java is compiled into bytecodes that allow it to run on any computer that supports the Java Virtual Machine. With Java you only need to write a program once and then you can run it on any machine that has a Java Virtual Machine.

## Creating, Compiling and Executing Java Programs

In this class, you will use a software package called IntelliJ IDEA to create, compile, and execute your programs. INTELLIJ IDEA is what is known as an **integrated development Environment**, or **IDE**.

IntelliJ IDEA contains a complete set of tools to create, compile, test, debug, and execute Java programs.

# Creating, Compiling and Executing the Program

## Source Code

Using the code editor in IntelliJ IDEA, we create the source code. The source code is written in Java.



## Creating, Compiling and Executing the Program



The compiler converts Java source code into an intermediate language called bytecodes.

## Creating, Compiling and Executing the Program



Java bytecodes are executed on a Java Virtual Machine.

The Java Virtual Machine (JVM) is a software program that runs on your computer. It *interprets* the byte codes by converting them into machine language code that is executed as it is converted.

## A Simple Java Program

```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```


Every java program must have at least one class. Each class has a name. By convention class names are capitalized. We will talk more about classes later in the course.

```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

Everything that belongs to the class is enclosed in these curly braces.



The keyword **public** is an access modifier. It tells us that this class can be seen by parts of the program code that are outside of this class.




```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

Every java program must also have exactly one method named main. Method names are not capitalized. The main method defines the entry point for the program, This is where the program will begin executing.

```
// This program displays the message
// Welcome to Java
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java");
    }
}
```

These are the *arguments* that are being passed to the main( ) method. We are not going to concern ourselves with these arguments, because the programs we write will never use them.

```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```



Methods are made up of statements. A statement is an instruction that tells the computer to do something. A statement ends with a semi-colon.

```
// This program displays the message
// Welcome to Java
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java");
    }
}
```

All of the statements that belong to a method are enclosed in curly braces. This is called the body of the method.



```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```


This statement tells the computer to display the string of characters inside the quotation marks. **System.out** is an object that represents the display screen on your computer. The display screen and the keyboard are often referred to as the computer's console.

```
// This program displays the message
// Welcome to Java
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java");
    }
}
```

**println** is the name of a method that outputs the string in the parenthesis to the display.

The text in between the quotation marks is known as a **string literal**.

Comments at the beginning of a program tell others what the program does. The characters `//` mark this as a single line comment.



```
// This program displays the message  
// Welcome to Java  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

## Programming Style

Programming style deals with what a program looks like. It is possible to write a program that works, but because of bad programming style it is hard to read, and therefore hard to maintain. Most software development organizations will require their programmers to follow a set of style guidelines when writing their source code.



# Naming Conventions

Choose meaningful and descriptive names.

**Class names:** Capitalize the first letter of each word in the name, for example,

WelcomeToJava.

This is called title case.

**Method names:** The first letter is not capitalized. For example,

computeArea()

This is called camel case.

## Indentation and Spacing

### Indentation:

everything inside of a block should be indented.

Normally Eclipse takes care of indentation for you.

### Spacing:

use a blank line to separate segments of the code.

# Programming Errors

- Syntax Errors
  - Detected by the compiler -the language has been used incorrectly.
- Runtime Errors
  - Cause the program to abort - the computer cannot do something you told it to do.
- Logic Errors
  - Produces incorrect results - the program works but there is something wrong in your logic

## Learning to program requires

- Time
- Patience
- Good language skills
- The ability to think abstractly
- Good math skills
- The ability to solve problems
- Practice – Program, program, program
- A sense of curiosity



## Learning to Program Takes Time

Researchers have shown that learning to do anything well (playing the piano, painting, playing tennis, etc) takes about 10 years. Learning to be a good programmer is no different.

To become proficient at programming

Practice

Practice

Practice ...

## Preparing For This Week's Assignment

The following slides illustrate how to download and install the Java JDK and IntelliJ IDEA. These slides show the install on a Windows computer, but installing on a Mac computer will be similar.

## The Java Development Kit (JDK)

The JDK contains a package of tools that are required for developing Java programs. You may already have the JDK installed on your computer, but you will want to download and install the most recent Long Time Support (LTS) edition. **This is currently JAVA SE 11.**

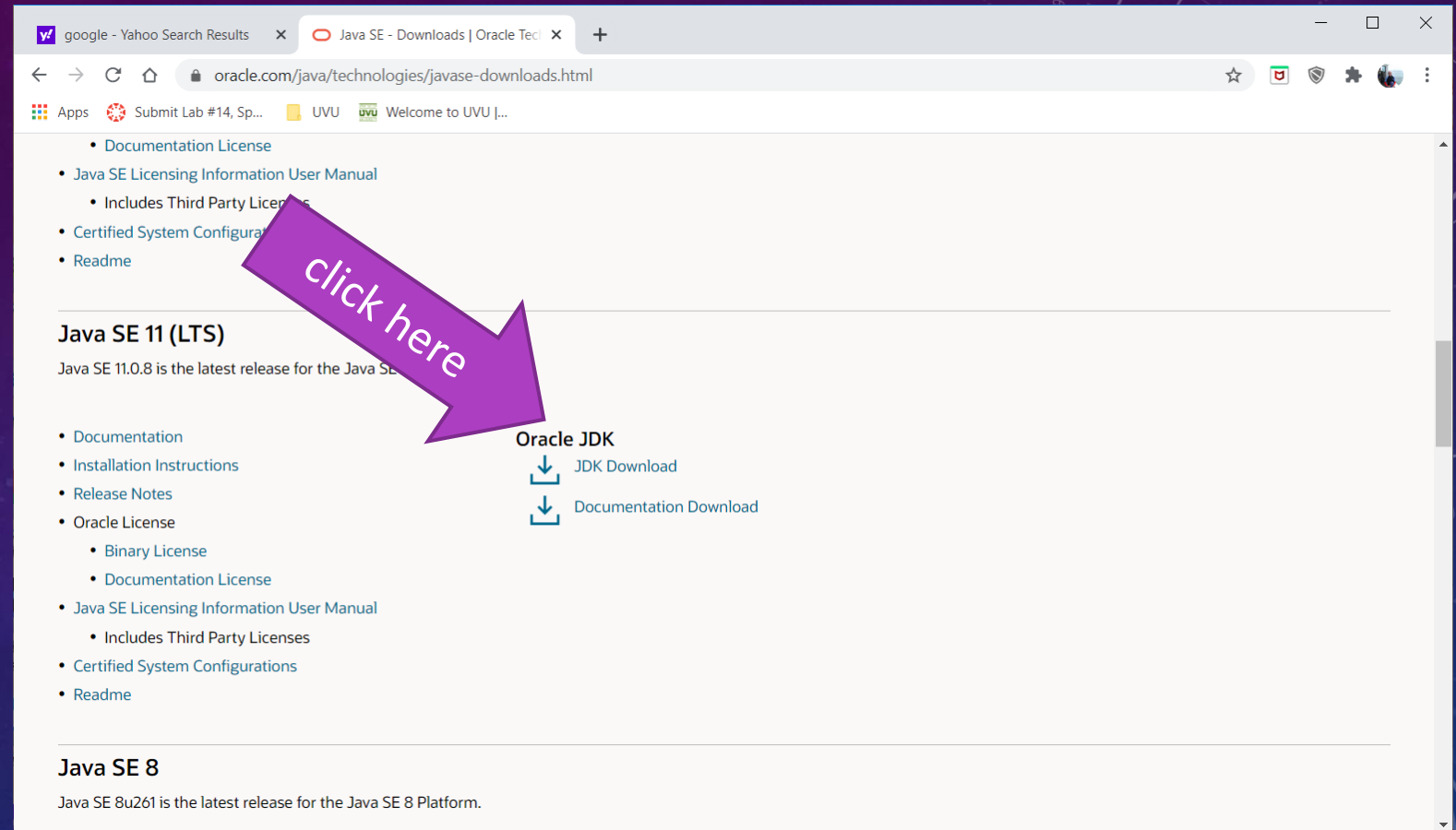
## Downloading and Installing The JDK

Downloads for the JDK are located at

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>









When you arrive at the landing page for the JDK downloads, scroll down until you find the information for Java SE 11 (LTS). This is the current long term support version of the JDK that you should download and use for this class. This is the current industry standard version.



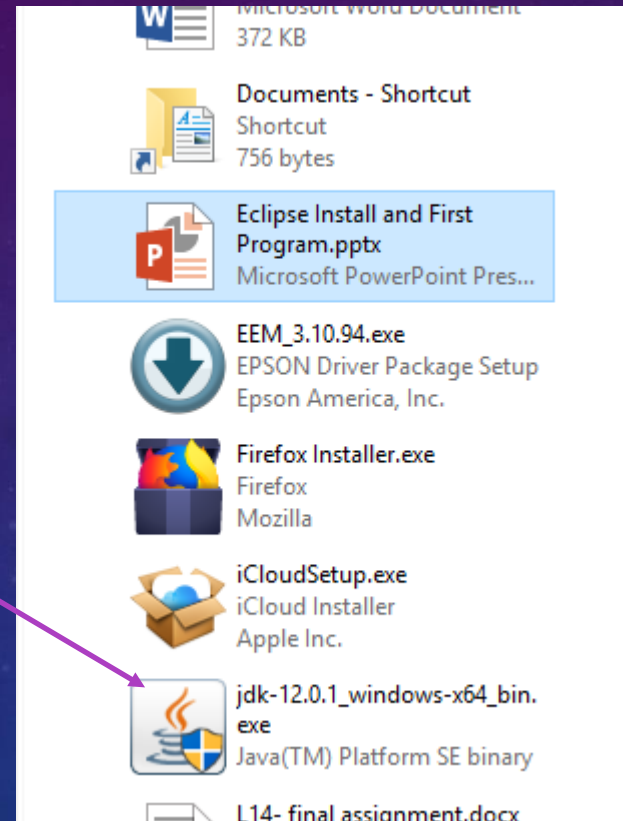
On this page, click on the download for your operating system.

The screenshot shows a web browser window with the address bar displaying 'oracle.com/java/technologies/javase-jdk11-downloads.html'. The page content includes a 'Java Magazine' link, a 'JDK 11.0.8 checksum' link, and a section titled 'Java SE Development Kit 11.0.8'. Below this title is a license notice: 'This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE'. A table lists the available download packages for different operating systems, including Linux (Debian, RPM, Compressed Archive), macOS (Installer, Compressed Archive), and Solaris SPARC Compressed Archive. Each row provides the product description, file size, and a download link with a download icon.

| Product / File Description       | File Size | Download  |
|----------------------------------|-----------|---|
| Linux Debian Package             | 148.77 MB |  <a href="#">jdk-11.0.8_linux-x64_bin.deb</a>            |
| Linux RPM Package                | 155.45 MB |  <a href="#">jdk-11.0.8_linux-x64_bin.rpm</a>            |
| Linux Compressed Archive         | 172.66 MB |  <a href="#">jdk-11.0.8_linux-x64_bin.tar.gz</a>         |
| macOS Installer                  | 166.84 MB |  <a href="#">jdk-11.0.8_osx-x64_bin.dmg</a>            |
| macOS Compressed Archive         | 167.23 MB |  <a href="#">jdk-11.0.8_osx-x64_bin.tar.gz</a>         |
| Solaris SPARC Compressed Archive | 186.49 MB |  <a href="#">jdk-11.0.8_solaris-sparcv9_bin.tar.gz</a> |

Find the downloaded file  
in your download folder  
and double-click on it to  
start the install process.

Follow the prompts.





Where did the install put the JDK?

Windows: C:/Program Files/Java/

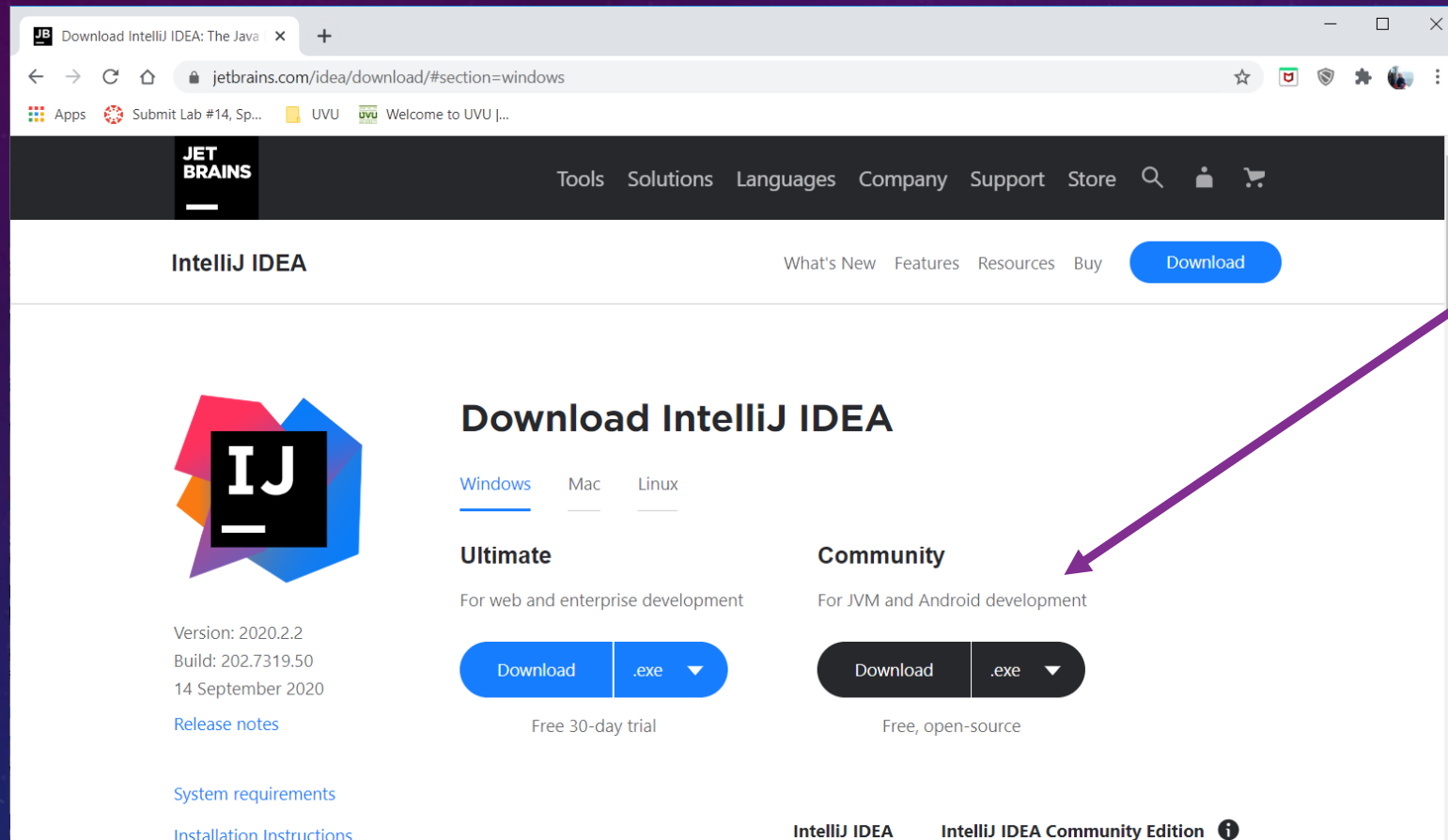
MacOS: Macintosh HD/Library/Java/JavaVirtualMachines/





Download and install IntelliJ IDEA

Go to <https://www.jetbrains.com/idea>



Download IntelliJ IDEA: The Java IDE

jetbrains.com/idea/download/#section=windows

Tools Solutions Languages Company Support Store

IntelliJ IDEA

What's New Features Resources Buy [Download](#)

**Download IntelliJ IDEA**

Windows Mac Linux

**Ultimate**  
For web and enterprise development

[Download](#) .exe

Free 30-day trial

**Community**  
For JVM and Android development

[Download](#) .exe

Free, open-source

Version: 2020.2.2  
Build: 202.7319.50  
14 September 2020

[Release notes](#)

[System requirements](#)

[Installation Instructions](#)

IntelliJ IDEA IntelliJ IDEA Community Edition

Download the free  
Community edition  
here.

ToolsLanguagesSolutionsSupportCompanyStore

IntelliJ IDEAComing in 2019.2What's NewFeaturesLearnBuyDownload



Version: 2019.1.3  
Build: 191.7479.19  
Released: May 27, 2019  
[Release notes](#)

[System requirements](#)  
[Installation Instructions](#)

# Download IntelliJ IDEA

WindowsmacOSLinux

## Ultimate

For web and enterprise development

DOWNLOAD.EXE

Free trial

## Community

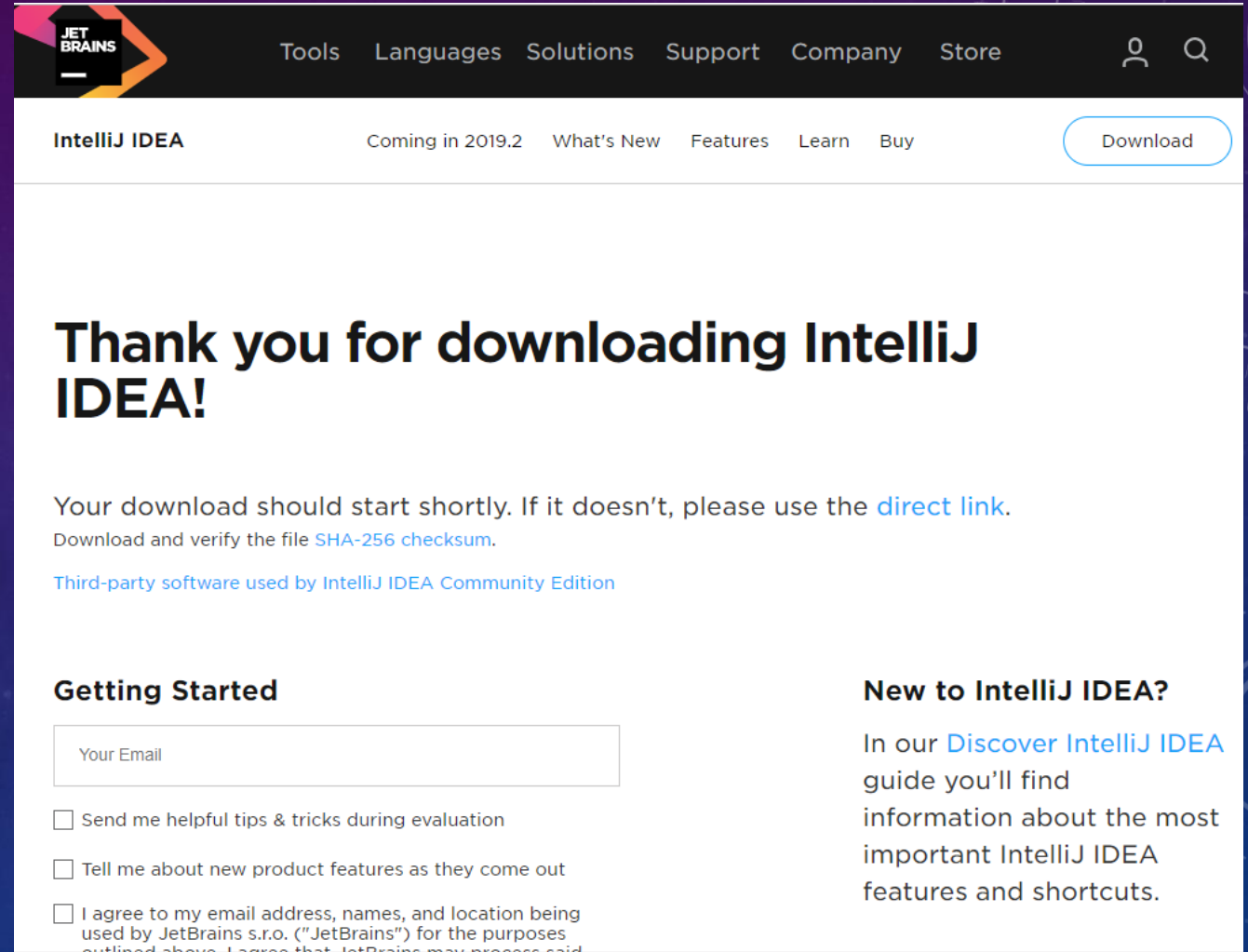
For JVM and Android development

DOWNLOAD.EXE

Free, open-source

Click on the  
Community  
download.

The download should start automatically. You do not need to provide an email unless you want to get emails from JetBrains.



The screenshot shows the IntelliJ IDEA download confirmation page. At the top, there is a dark navigation bar with the JetBrains logo on the left and links for Tools, Languages, Solutions, Support, Company, and Store on the right. Below this is a white header bar with 'IntelliJ IDEA' on the left, 'Coming in 2019.2', 'What's New', 'Features', 'Learn', and 'Buy' in the center, and a 'Download' button on the right. The main content area has a large heading 'Thank you for downloading IntelliJ IDEA!' followed by instructions: 'Your download should start shortly. If it doesn't, please use the [direct link](#). Download and verify the file [SHA-256 checksum](#). [Third-party software used by IntelliJ IDEA Community Edition](#)'. Below this is a 'Getting Started' section with an email input field and three checkboxes: 'Send me helpful tips & tricks during evaluation', 'Tell me about new product features as they come out', and 'I agree to my email address, names, and location being used by JetBrains s.r.o. ("JetBrains") for the purposes outlined above. I agree that JetBrains may process said...'. On the right side of the 'Getting Started' section is a 'New to IntelliJ IDEA?' section with the text: 'In our [Discover IntelliJ IDEA](#) guide you'll find information about the most important IntelliJ IDEA features and shortcuts.'

**IntelliJ IDEA** Coming in 2019.2 What's New Features Learn Buy [Download](#)

## Thank you for downloading IntelliJ IDEA!

Your download should start shortly. If it doesn't, please use the [direct link](#).  
Download and verify the file [SHA-256 checksum](#).  
[Third-party software used by IntelliJ IDEA Community Edition](#)

### Getting Started

☐ Send me helpful tips & tricks during evaluation

☐ Tell me about new product features as they come out

☐ I agree to my email address, names, and location being used by JetBrains s.r.o. ("JetBrains") for the purposes outlined above. I agree that JetBrains may process said...

### New to IntelliJ IDEA?

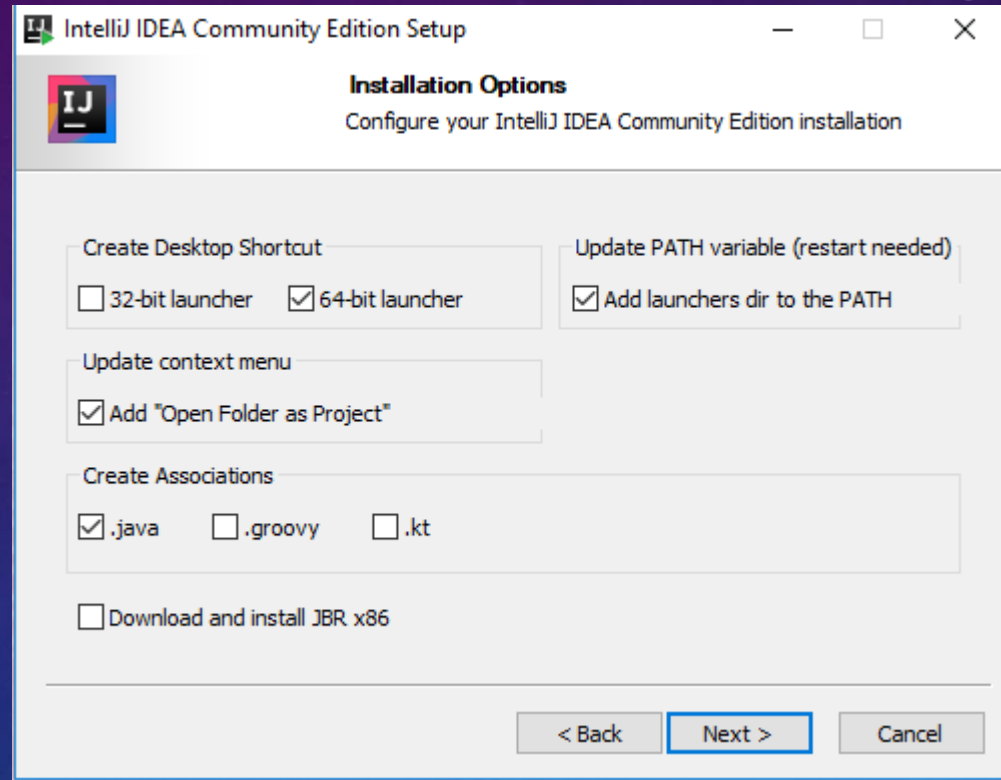
In our [Discover IntelliJ IDEA](#) guide you'll find information about the most important IntelliJ IDEA features and shortcuts.

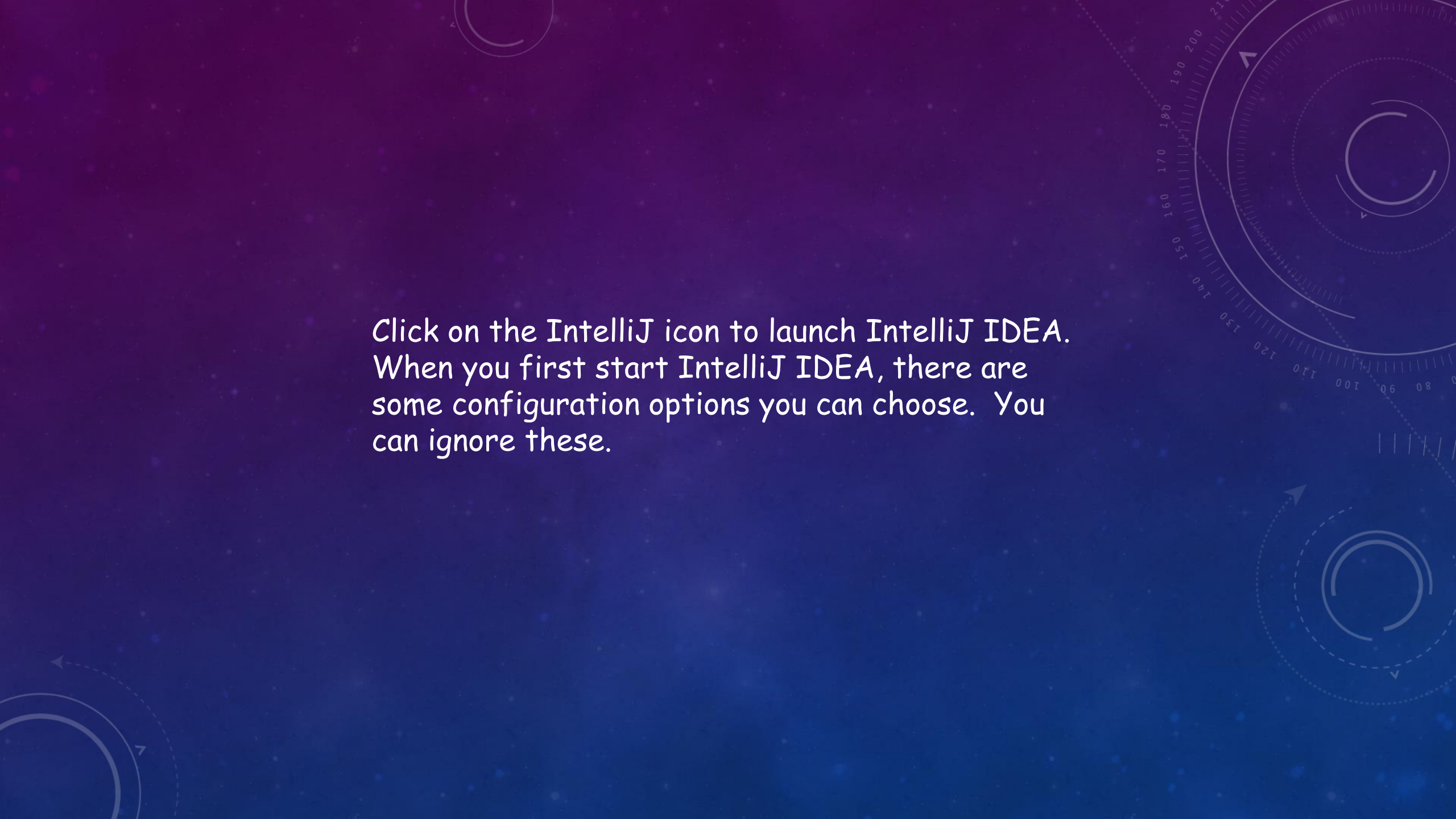


Double click on the downloaded file to start the install process. Follow the prompts.



Select these options.  
(Windows only):





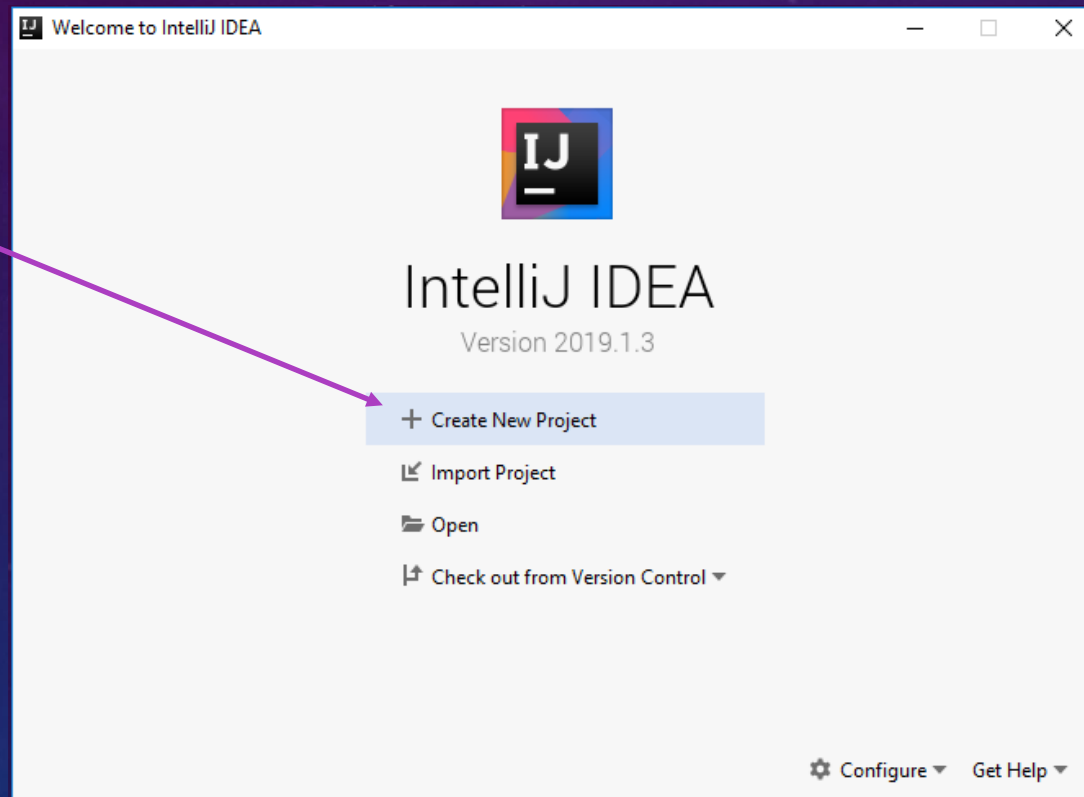
Click on the IntelliJ icon to launch IntelliJ IDEA.  
When you first start IntelliJ IDEA, there are  
some configuration options you can choose. You  
can ignore these.

# Writing Your First Java Program

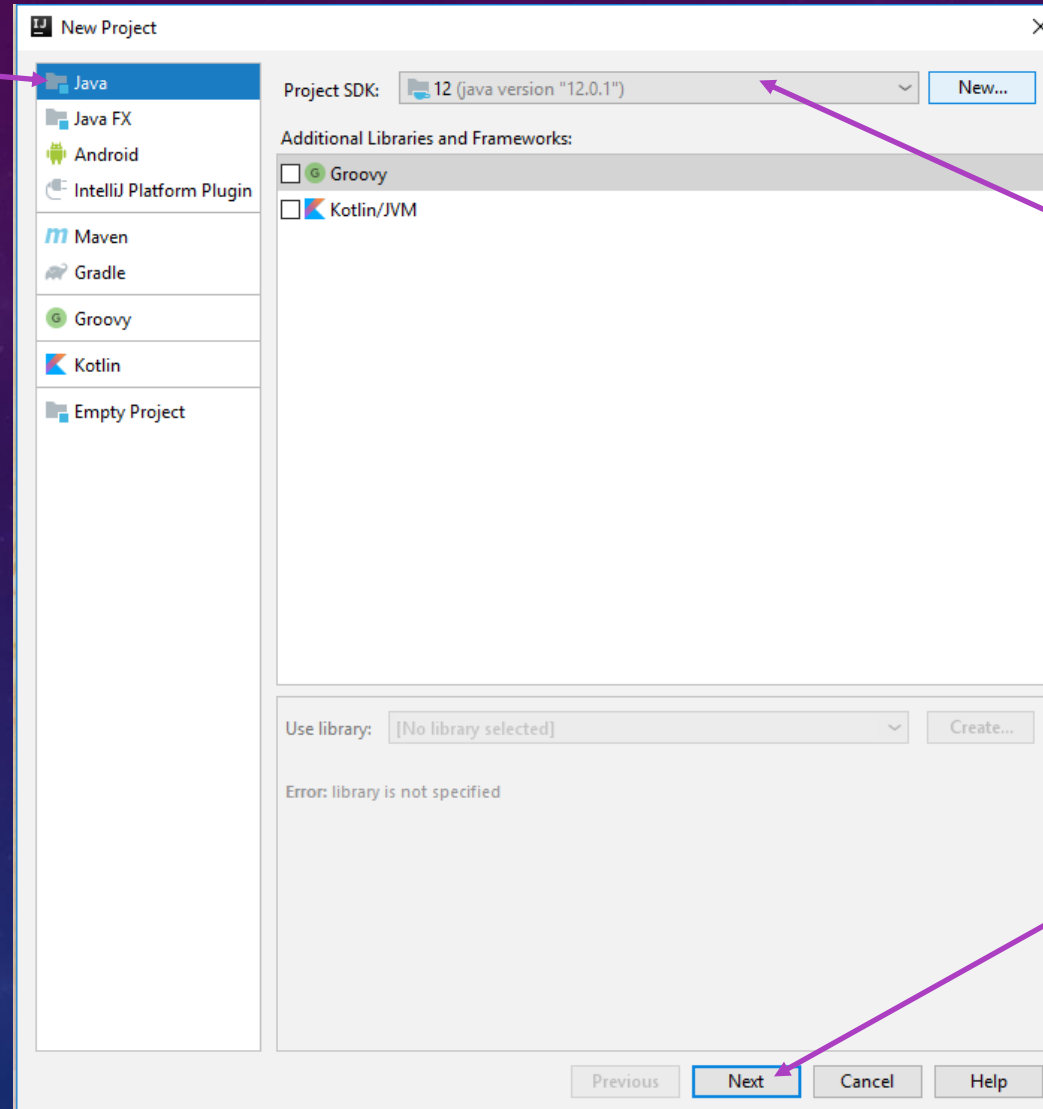




Click on Create  
New Project



Select Java



If the JDK you installed does not show up here, click on the **New** button and find the installed JDK

Click on Next.

Click on the Create project from template box

Pick Command Line App

New Project

☒ Create project from template

Command Line App

Simple Java application that includes a class with main() method

Previous Next Cancel Help

Click on Next

Type in a project name

Use the default location  
or type a different location.

Type a package name

Click on  
Finish

New Project

Project name: Proj01

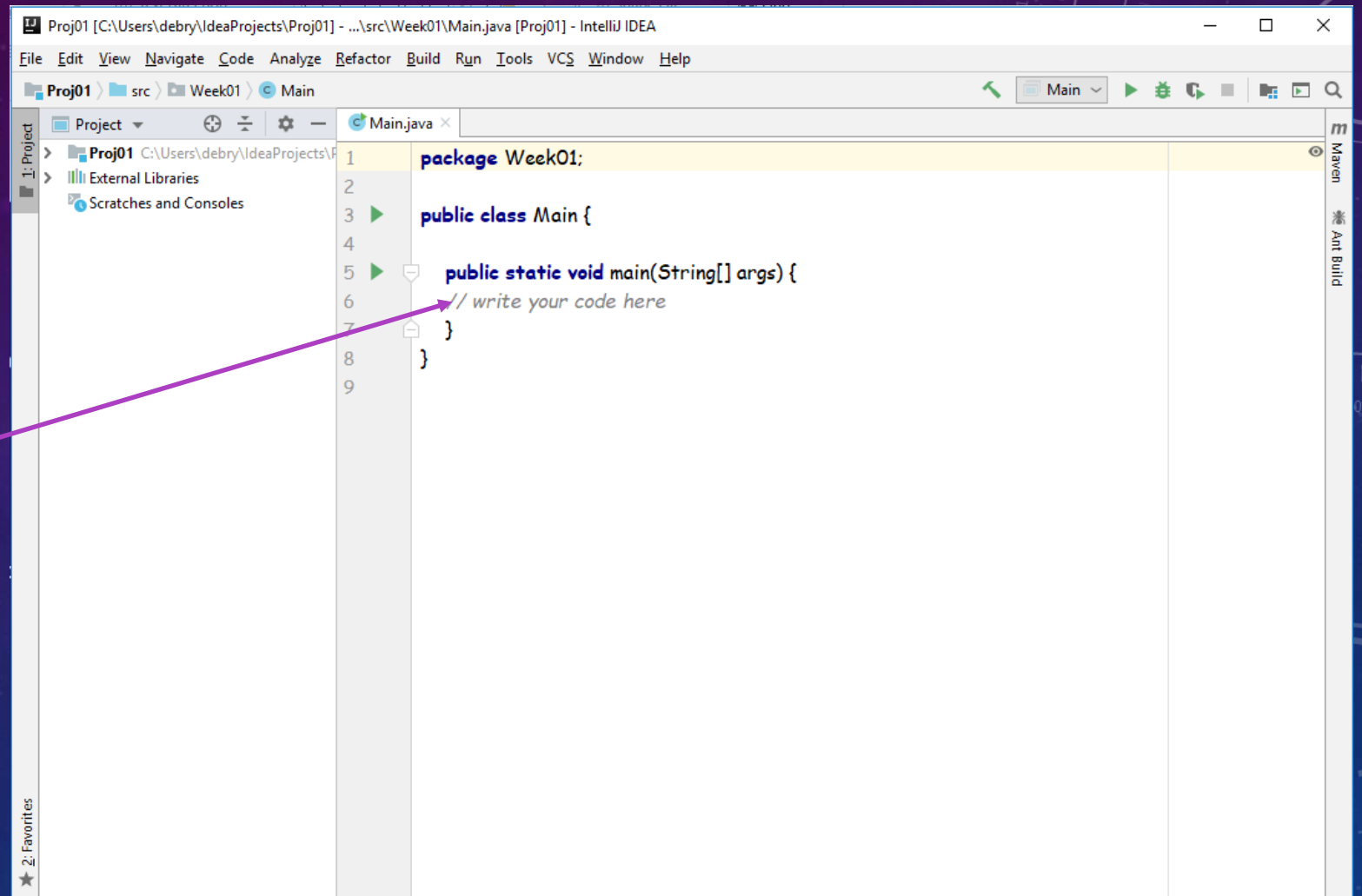
Project location: C:\Users\debry\IdeaProjects\Proj01

Base package: Week01

Previous Finish Cancel Help

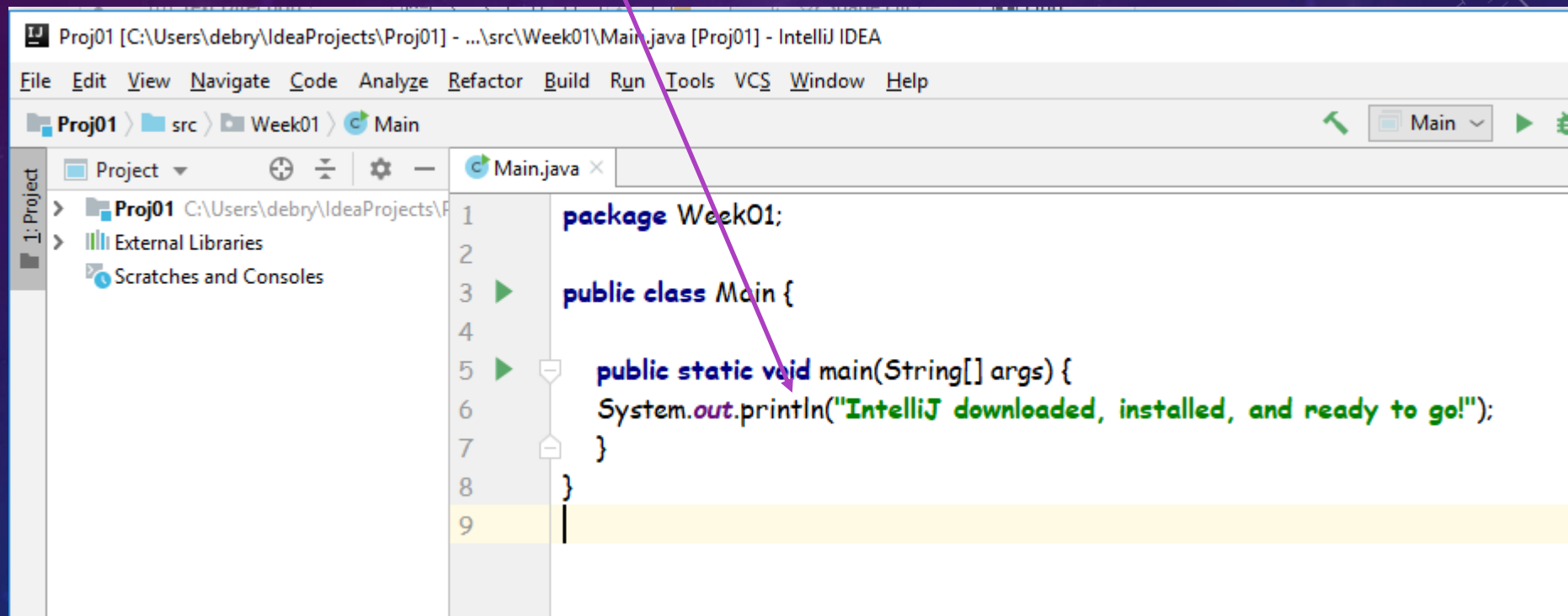


Add your code here



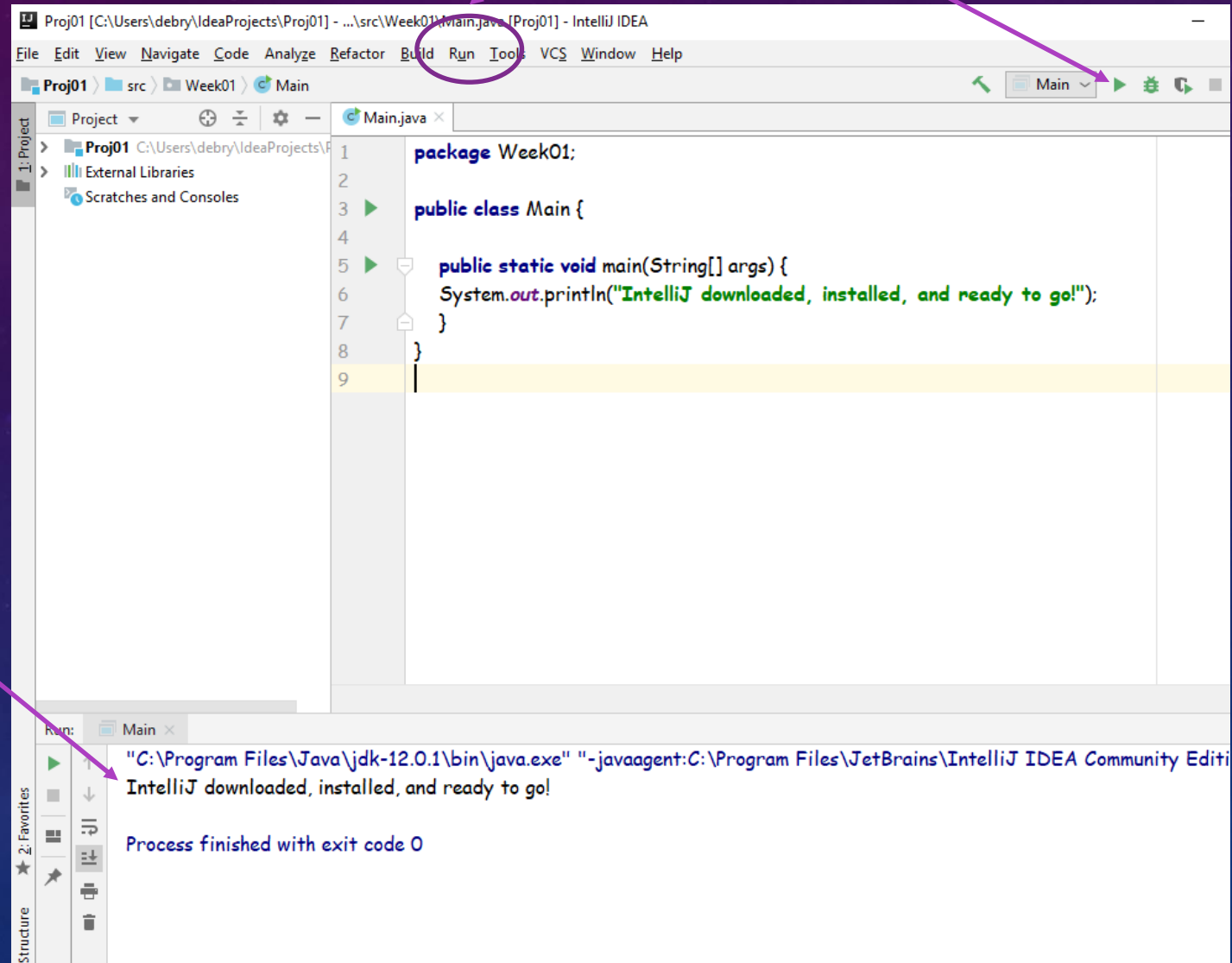
```
1 package Week01;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // write your code here
7     }
8 }
9
```

Add this line of code



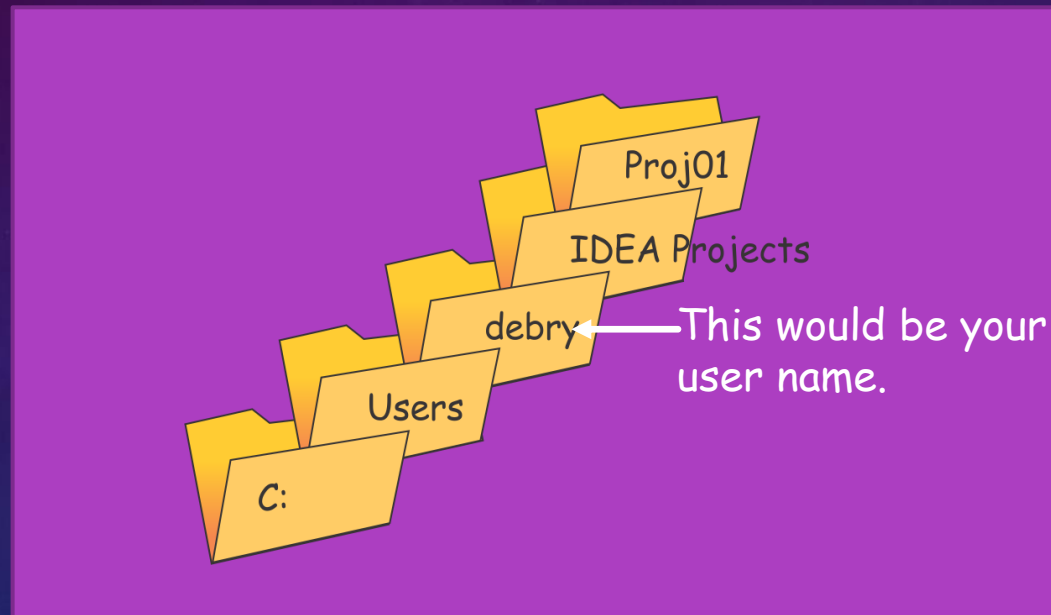
```
1 package Week01;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("IntelliJ downloaded, installed, and ready to go!");
7     }
8 }
9
```

Click on Run->Run Main on the menu bar or click on this green triangle to run your program.



The output of your program appears here

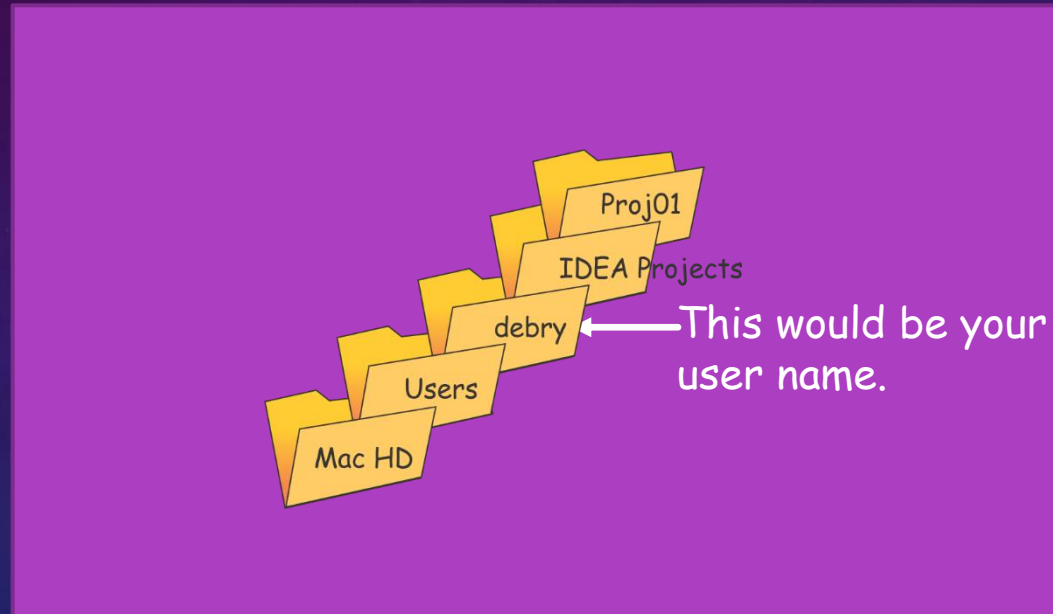
## Where are my files (Windows)?



If you use the default location for your files, your program will be in a folder with the project name, in a directory tree like this:



## Where are my files (Mac)?



If you use the default location for your files, your program will be in a folder with the project name, in a directory tree like this:

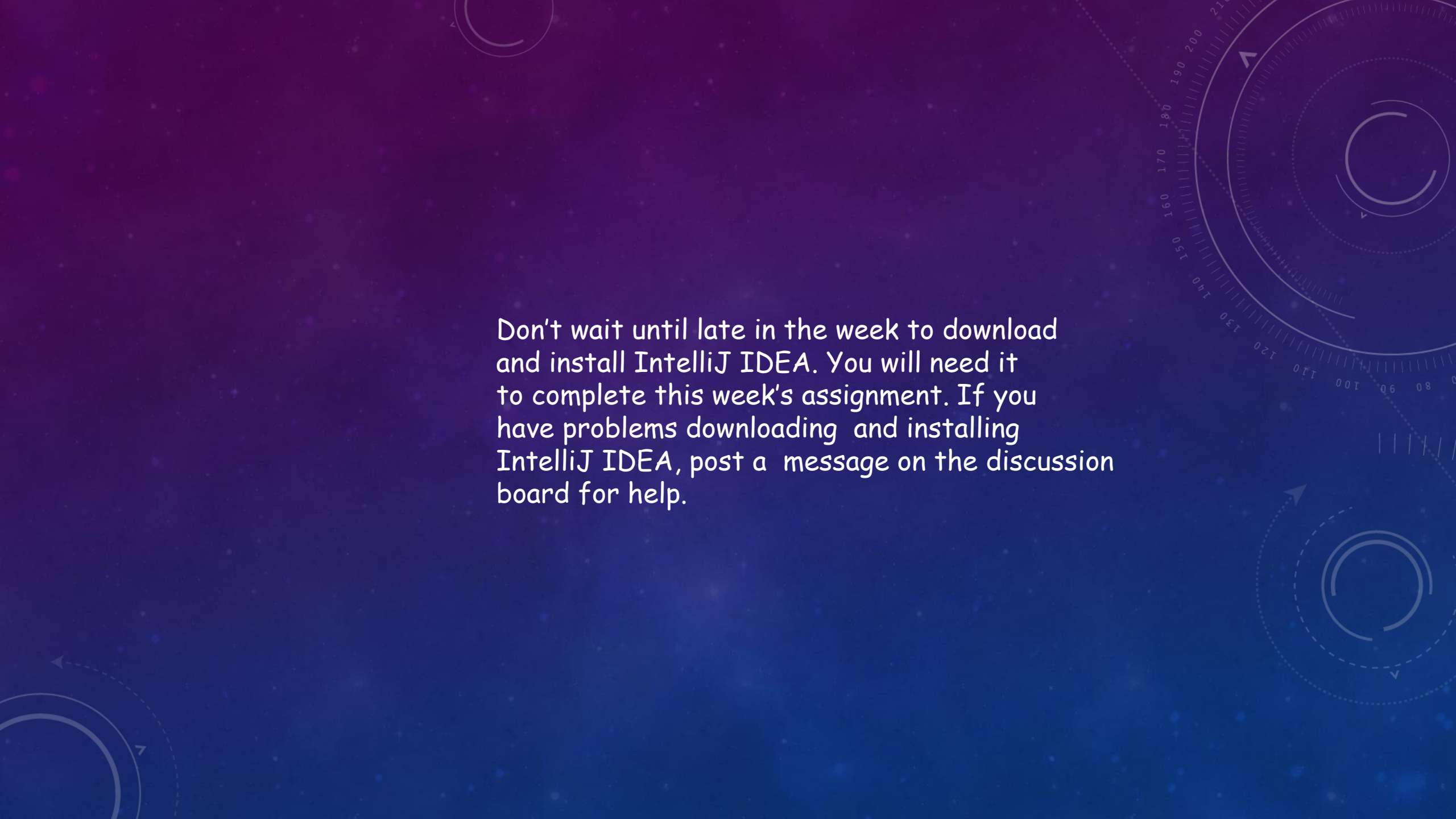
Video presentation on running IntelliJ  
IDEA for the first time

[https://youtu.be/c0efB\\_CKOYo](https://youtu.be/c0efB_CKOYo)

## To Submit Your Program

- (1) Create a zip folder
- (2) Drag the entire project folder and drop it into the zip folder
  - If the assignment has two programs, you will have two project folders
- (1) Submit the zip folder





Don't wait until late in the week to download and install IntelliJ IDEA. You will need it to complete this week's assignment. If you have problems downloading and installing IntelliJ IDEA, post a message on the discussion board for help.