

Introduction to Java

Java Accelerator 7

Lesson 1.1



Introduction and Level Set

Purpose

The purpose of this lesson is to ensure that:



You've set up your development environment properly.



You've set up your Jira account properly, and you understand how Agile tools and concepts will be used in this course.



You understand non-object-oriented Java language syntax and concepts and how they compare to other languages.

Learning Objectives



Create, debug, and compile Java terminal applications.



Write simple applications that utilize core Java types and syntax.



Use Maven to compile and run terminal applications.



Use planning meetings to plan work.



Use Jira to track work.

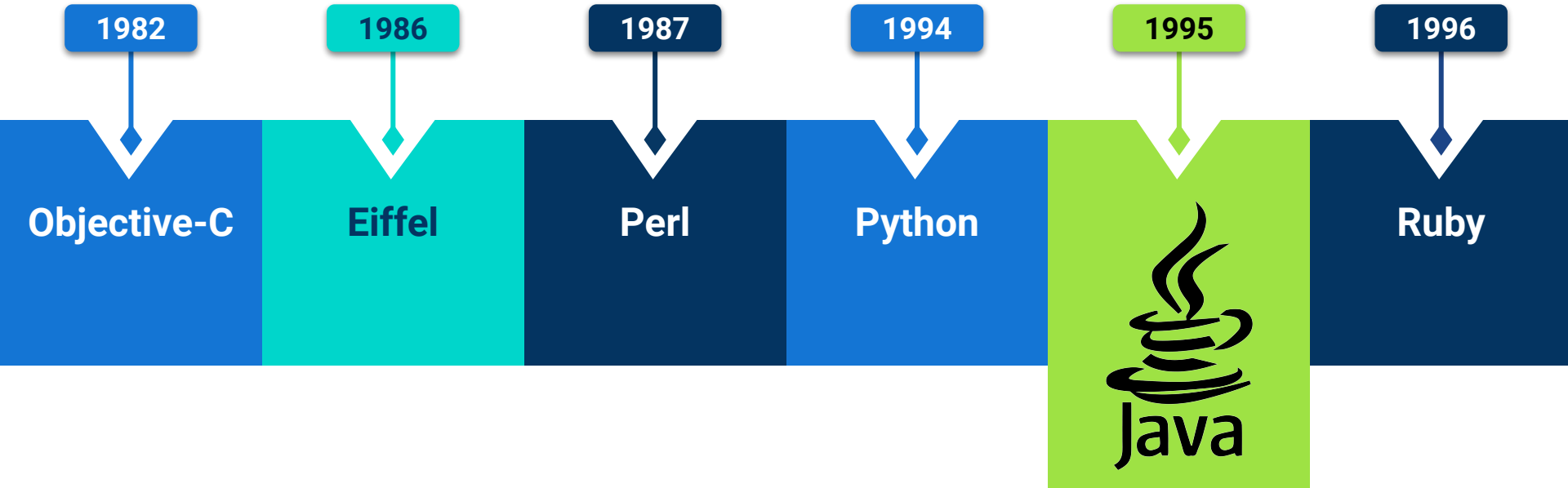


Participate in retrospectives.

Java Tools of the Trade

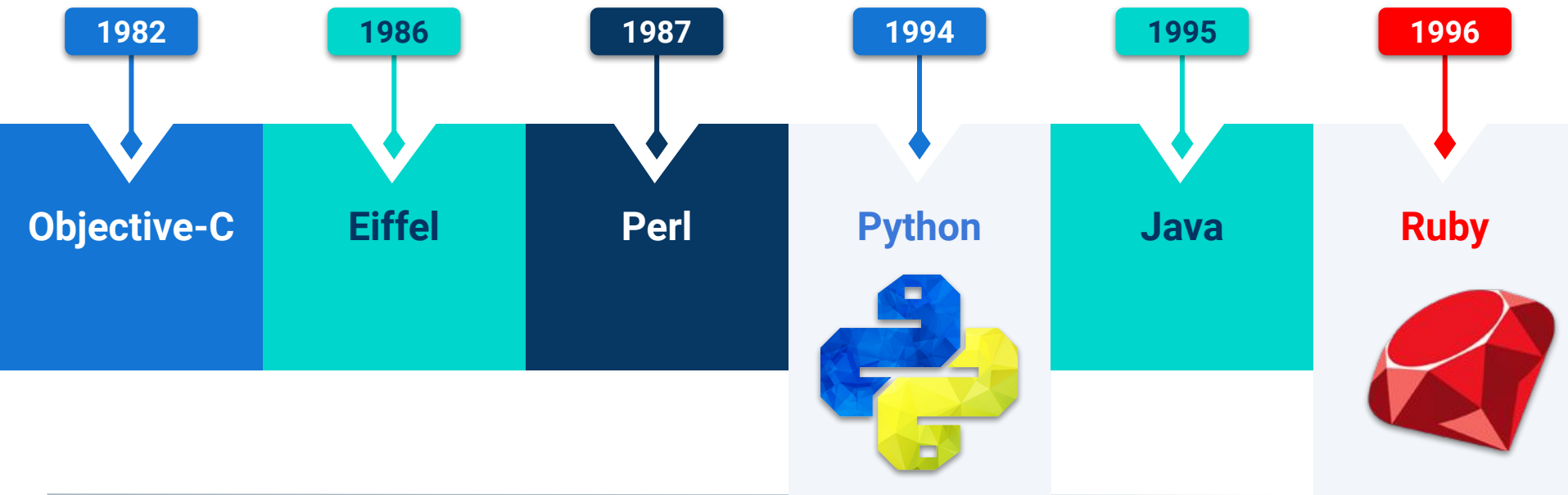
The History of Java

Java was created in 1995 by James Gosling, who worked at Sun Microsystems.



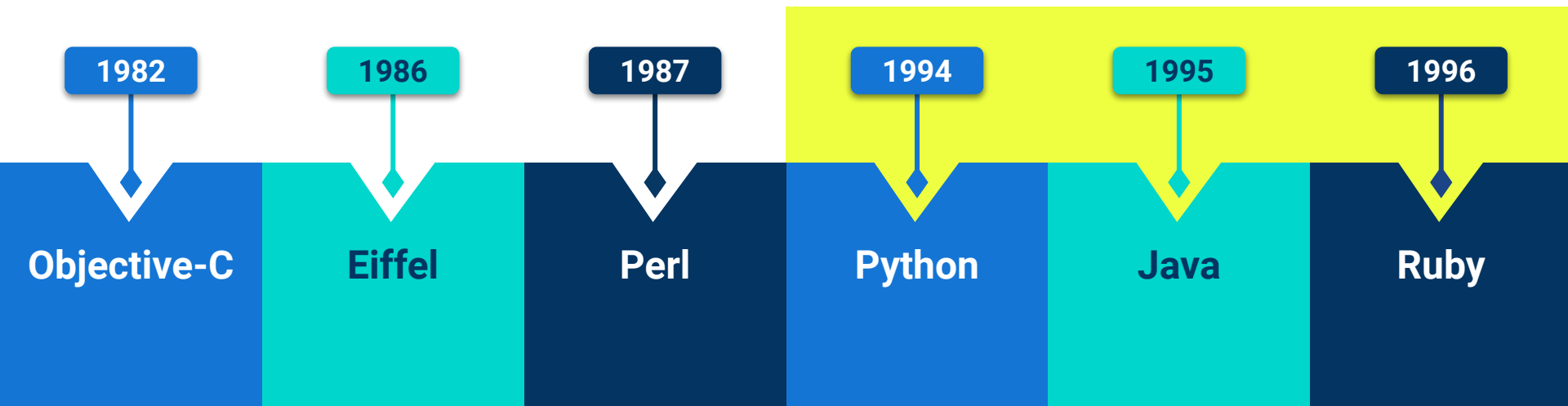
The History of Java

Python and Ruby were released around the same time: Python in 1994 and Ruby in 1996.



The History of Java

Java is often considered old and stodgy compared to Python and Ruby (which are considered more hip), but they were all released around the same time.



The History of Java

Sun was purchased by Oracle in 2010. Sun ceased to exist, and Oracle became the steward of Java.



COVID-19

BEST PRODUCTS ▾

REVIEWS ▾

NEWS ▾

HOW TO ▾

FINANCE ▾

CARS ▾

DEALS ▾

Oracle buys Sun, becomes hardware company

The last chapter for Sun Microsystems closes, and the next for Oracle begins as the software company adds hardware to its portfolio.

The History of Java

Sun made Java open source in 2006. Until then, Java had always been free, but it had not been open source.

The open source Java project is called **OpenJDK**.

OpenJDK



What is this? The place to collaborate on an open-source implementation of the [Java Platform, Standard Edition](#), and related projects. ([Learn more.](#))



Download and [install](#) the open-source JDK for most popular Linux distributions. Oracle's free, GPL-licensed, production-ready OpenJDK JDK 14 binaries are at [jdk.java.net/14](#); Oracle's commercially-licensed JDK 14 binaries for Linux, macOS, and Windows, based on the same code, are [here](#).

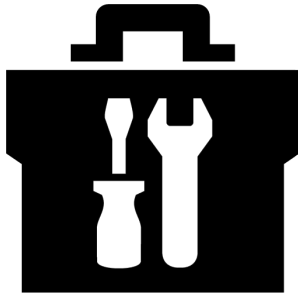
Java Tools and Platforms

Java Tools and Platforms

Java Development Kit (JDK) and Java Runtime Environment (JRE)

Java Development Kit (JDK)

Contains the tools necessary for developers to create and run new Java programs.



Java Runtime Environment (JRE)

Allows people to run Java programs that others have written.





The JDK contains the JRE!

Oracle Java vs. OpenJDK

Oracle Java

- Commercially supported.
- Released, distributed, and supported by Oracle.

vs.

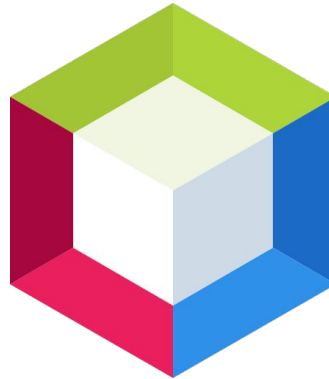
OpenJDK

- Open source project from which Oracle Java is derived.

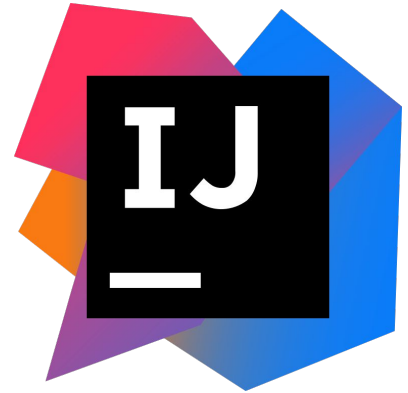
Integrated Development Environment (IDE)

The three most popular Java IDEs are Netbeans, Eclipse, and IntelliJ.

Many companies let developers choose their preference; some companies standardize on one.



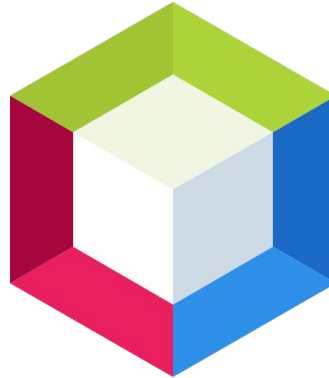
NetBeans



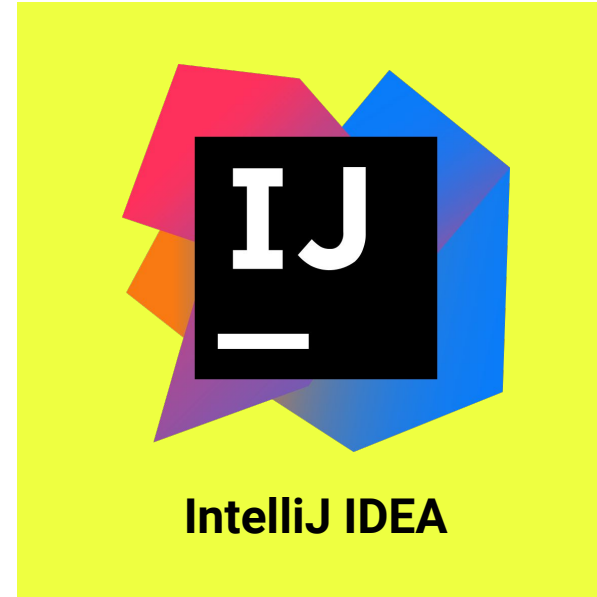
IntelliJ IDEA

Integrated Development Environment (IDE)

We will be using IntelliJ IDEA in the course.



NetBeans



Libraries and Frameworks

Java developers use a variety of libraries and frameworks to build applications, many of which are open source.

01

In this course we'll use Apache Maven as a build and dependency management tool.

02

In addition, we'll use Apache Tomcat as our local web container server.

03

Spring Framework is a very important Java framework that we'll use throughout this course. Spring Boot helps us get Spring Framework projects started quickly.



Time to Code



Hello, World!

Suggested Time:

10 minutes

Hello, World Code

```
package com.company;

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

Jira and Agile Concepts



Time to Code

Jira

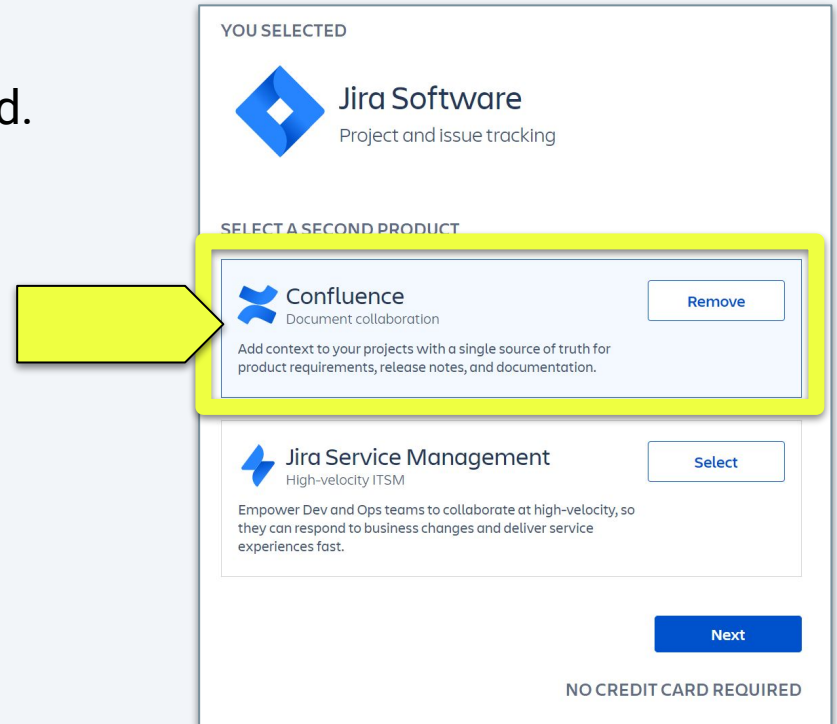
Suggested Time:

20 minutes


Account Creation (1 of 2)

Visit the Jira sign-up page at <https://www.atlassian.com/software/jira/free>.


Select Confluence as a second product along with Jira, then select Next to proceed.




YOU SELECTED

 **Jira Software**
Project and issue tracking

SELECT A SECOND PRODUCT

 **Confluence**
Document collaboration
Add context to your projects with a single source of truth for product requirements, release notes, and documentation. [Remove](#)

 **Jira Service Management**
High-velocity ITSM
Empower Dev and Ops teams to collaborate at high-velocity, so they can respond to business changes and deliver service experiences fast. [Select](#)

[Next](#)


NO CREDIT CARD REQUIRED

Account Creation (2 of 2)

Use your personal email address in the sign-up process.
Select Agree to proceed.

Get started

Free for up to 10 users

 Continue with Google

OR

Work email *

john_doe@johndoe.com

Password *

.....

Password Strength: Very strong

First name *

John

Last name *


Doe

By clicking below, you agree to the [Atlassian Cloud Terms of Service](#) and [Privacy Policy](#).

Agree

NO CREDIT CARD REQUIRED

This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.

 ATlassian

Jira Setup

When prompted, indicate that you:

01

Are new to Jira.

02

Are new to agile methodologies.

03

Spend time working on features.

04

Have a tight schedule to finish work.

Help us set up your Jira

I am new to Jira.

My team is new to agile methodologies.

We spend our time working on features.

We have a tight schedule to finish our work.

Skip

Next

Jira Project Creation

When prompted, choose Scrum for the project template and “Team-managed” for the project type.

For the project name, use your first and last name, followed by “Course Work”.

Add project details


You can change these details anytime in your project settings.

Name

Key ⓘ

Template

Change template




Scrum RECOMMENDED

Sprint toward your project goals with a board, backlog, and roadmap.

Type

Change type



Team-managed RECOMMENDED

Control your own working processes and practices in a self-contained space.

[Back](#) [Create project](#)

Organizing Work - Stories

Issue	An issue is the basic unit of work for an agile project in Jira. We'll use the story type most of the time in this class.
Story	A story is something that provides demonstrable business value to the project's customer.
Customers	In this course, your customers are your instructional staff. The value you are delivering is the verification that you are picking up the required skills.
Tasks	A story might have multiple tasks . These tasks collectively deliver the business value to the customer.



Epics are a way to group related stories.

For course work, each lesson will be an epic, and each exercise/assignment will be an issue of the story type.

Backlog and Sprints

Backlog

The **backlog** is the place where you prioritize upcoming issues. Simply drag the issues with the highest priority to the top of the list.

Sprint

A **sprint** is a timebox of a particular length that teams use as the cadence or rhythm of their project.

The sprint length for course work is one week.

You will be working on some bigger projects at certain points in the course. These might be broken down into multiple, shorter sprints.

Issue Workflow

Issues go through three states as you work on them:

To Do

The initial state of a newly created issue.

In Progress

Change the state to In Progress when you begin working on an issue.

Done

Change the state to Done when you are finished working on the issue and want your instructional staff to look at it.



Time to Code



Our First SPM

Suggested Time:



Activity: Basic Java

- A Chatty Program
- A Postcard from Me

Suggested Time:

Java Statements, Expressions, Types, and Variables



Java Keywords

[https://docs.oracle.com/javase/tutorial/java/nutsandbolts/ keywords.html](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/keywords.html)

Data Types

The three most commonly used Java data types:

Strings	text
Numbers	whole numbers and decimals
Booleans	true or false

Data Types in Code

```
public class MyClass {  
    public static void main(String[] args) {  
        int wholeNumber = 5; // integer  
        float decimalNumber = 5.99f; // floating point number  
        boolean trueStory = true; // boolean  
        boolean bigLie = false; //  
        String textContent = "Strings contains text content.";  
        System.out.println(textContent);  
        System.out.println(wholeNumber);  
        System.out.println(decimalNumber);  
        System.out.println(trueStory);  
        System.out.println(bigLie);  
    }  
}
```

Variables

The overall concept of a variable in Java is the same as it is in other languages.

The big difference is that a Java variable declaration requires **both a type and a name**.

Java uses the same assignment operator (=) to assign values to variables. Remember that Java is a **statically typed language**. This means that the type of the variable you specify at declaration can never change.

```
String team = "Miami Dolphins";
```

Identifier Rules

Keywords cannot be used as identifiers.

Identifiers cannot span multiple lines. This makes sense—why would you even want to do this?

Identifiers can only contain the following characters:

```
Numbers  
Letters  
Underscores (_)  
Dollar signs ($)
```

Identifiers must start with a letter, underscore, or dollar sign—in other words, they can't start with numbers. Also, they cannot contain spaces.

Naming Conventions

The first letter in variables and method names are lowercase.

The first letter in a class name is uppercase.

Java uses **camelCase** for naming:

```
myTeam (variable)
calculateCustomerTotal() (method)
StudentRecord (class)
```

Names should be descriptive. Classes and variables are nouns, while methods are verbs.

An **expression** is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

–Oracle Java Documentation



Operators

01

Arithmetic, assignments, and comparison operators in Java are similar to those in other languages.

02

Java has the assignment operator `(=)` and the comparison operator `(==)` only.

03

Java also has the `.equals()` method for comparing the equality of objects.

Comments

Java comment syntax is similar to that in other languages.

```
/* This is  
a multi-line comment */  
System.out.println("Hello World");  
  
System.out.println("Hi!"); // This is an inline comment
```

Console IO

The Scanner

There are several ways to get console input from the user. We'll use something called the Scanner.



We use the Scanner because it is similar to `System.out.println(...)`.



We can use it to read from and write to files later on.



Time to Code



Hello and Add

Suggested Time:

Debugging

Why Do We Use the Debugger?

01

The debugger allows us to observe what is actually going on in a program vs. what we think is going on.

02

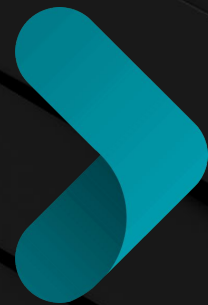
The debugger is one of the most important tools in a developer's tool belt, but many beginners do not take advantage of it and suffer for it.

03

Learn to use the debugger well and it will save you countless hours over your career, and maybe even in this class.



Time to Code



Debugging

Suggested Time:



Activity: Exercises

- Rectangular Paving Company
- Command Line Gram

Suggested Time:

Controlling Program Flow

Boolean Expressions & Operators

Boolean expressions and operators in Java are similar to those in other languages, except for truthiness.

Boolean expressions in Java use only the boolean data type.

```
boolean story = true;  
boolean bigLie = false;
```

Relational Operators

Relational operators in Java are similar to those in other languages.

Java has no such operator; it has only double equals `(==)`, used for comparison of primitive types.

Java uses `.equals()` to compare two objects of the same type.

Equality	<code>(==)</code>
Greater than, less than	<code>(>, <)</code>
Not equal	<code>(!=)</code>
Greater than or equal, less than or equal	<code>(>=, <=)</code>

Conditional Statements

Java and other languages share similar sets of conditional statements:

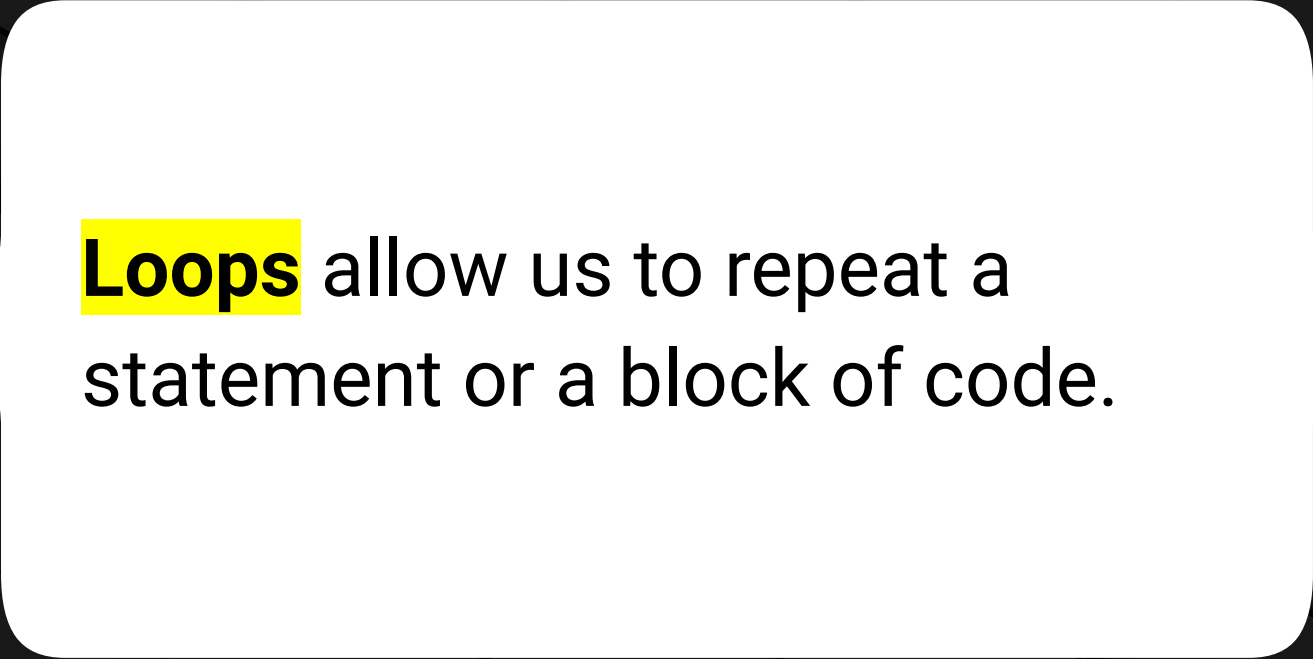


if-else



switch

```
if (condition) {  
    // execute code if condition  
    // is true  
}
```



Loops allow us to repeat a statement or a block of code.

Loops

Java has the following loop constructs:

for	Similar to other languages.
for-each	This is similar to other languages and is used with the data structures and objects that we'll encounter later in the course.
while	Similar to other languages.
do...while	Similar to other languages.



Activity: Controlling Program Flow

- Fizz Buzz
- Prime Finder

Suggested Time:

Methods

Methods Overview



Methods are similar to functions in other languages in form and in what they do.

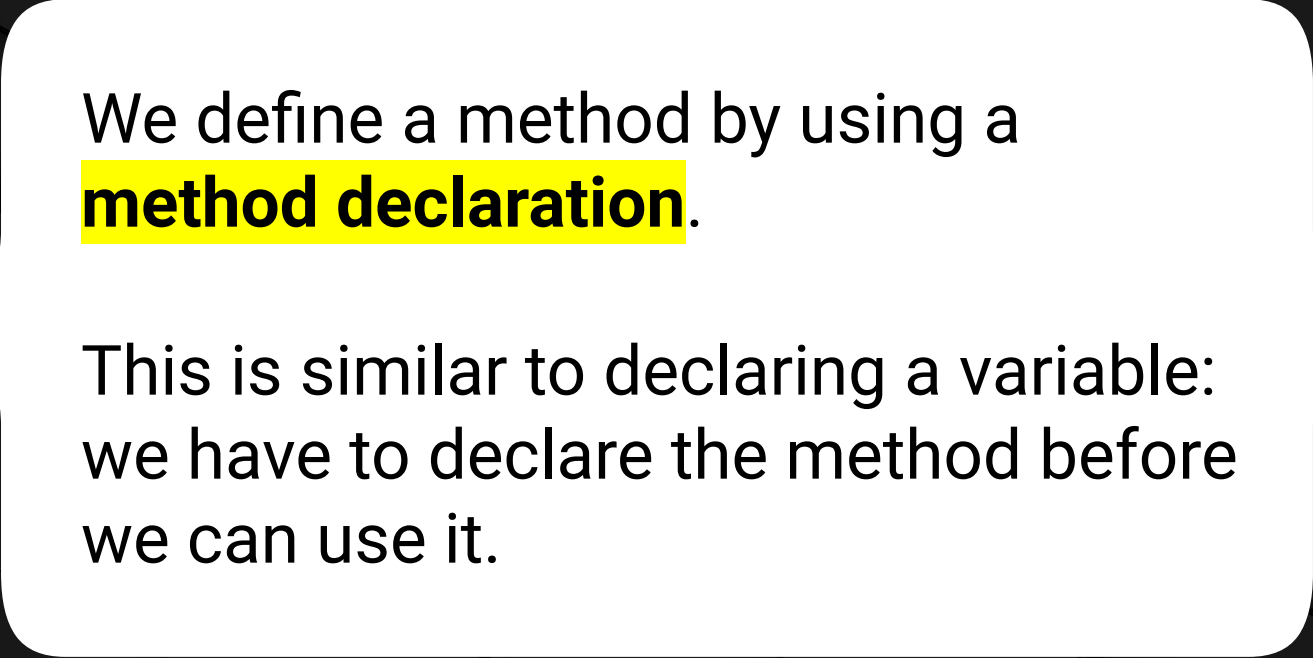


Java methods must be associated with an object.
There is no concept of a global function that is not attached to an object.



The closest thing that Java has to global functions are static methods that can be used without instantiating the object with which they are associated.

Defining Methods



We define a method by using a **method declaration**.

This is similar to declaring a variable:
we have to declare the method before
we can use it.

Method Declaration

The method declaration has several parts. We'll start with the basics and build outward. The main pieces are:



The name of the method.



The body (code block) of the method.



Any parameters that the method takes. (More on this later.)



The type of data returned by this method. (More on this later too.)



Parameters and return type.

Full Method Description

```
<access modifier> <return type> <method name>  
(<parameter list>) <exception list> {  
    <method body>  
}
```

Method Example

```
public static int getMaximum(int x, int y) {  
    if (x > y) {  
        return x;  
    } else {  
        return y;  
    }  
}
```

Method Signature

The **method signature** is what we use to uniquely identify methods. It consists of:



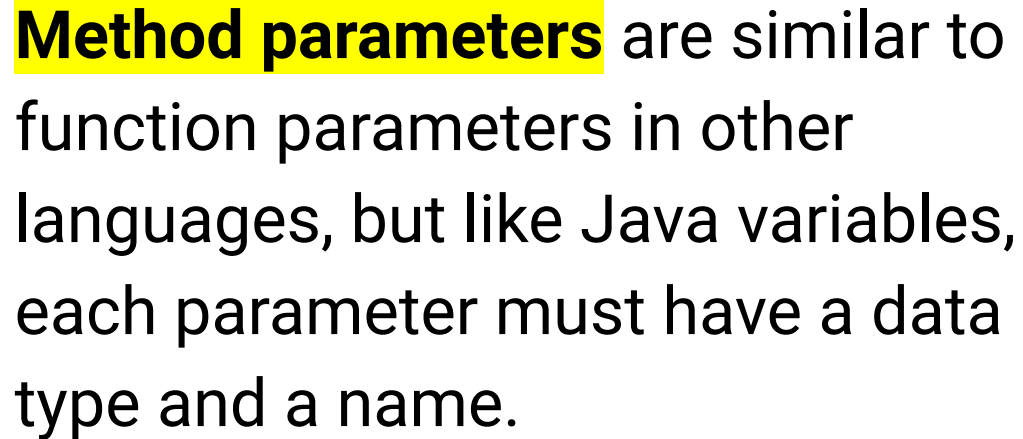
Method name



Parameter list

Example:

```
greaterValue(int x, int y)
```



Method parameters are similar to function parameters in other languages, but like Java variables, each parameter must have a data type and a name.

Return Types and Access Modifiers

Return Types

You must specify the data type that is returned by a method.

The return type for a method that returns no value is `void`.

Access Modifiers

For now, we'll just be using `public`. We'll learn the meaning of access modifiers in future lessons.

Method Naming Conventions



Begins with a lowercase letter.



Is a verb (if a single-word name).



Begins with a verb (if a multi-word name) followed by other words.



Should be in camelCase (the first letter of the second and following words should be capitalized).

Example:

```
calculateTotal  
processOrders  
storeUserData  
checkCardValidity
```

Static Keyword

We're not going to explain `static` yet—we'll save that for the object-oriented part of the course.



The `static` keyword is necessary for all methods for the moment.

Scope

Member Variables and Class-Level Scope

These variables are declared inside of a class (yet outside of any method) and can be accessed anywhere inside the class.

The following modifiers can be added to the variable to further refine their scope:



public



protected



private



A method or variable without any of the above has “default” access.

Scope

Local Variables / Method-Level Scope

Declared inside a method and not usable outside of it.

Loop Variables / Block Scope

Declared inside a block and not usable outside of it.



Activity: Methods

- Method Practice

Suggested Time:

Arrays

Arrays Overview

Java arrays are similar to arrays in other languages but are much less flexible and powerful.

Java has other data structures that have the power and flexibility of arrays in other languages—we will talk about them shortly.

Example:

```
String[] fruits = {"apples", "oranges", "bananas"};
```


Declaring Arrays

Arrays must be declared and initialized just like the variables that we've already encountered.

The difference is that arrays are made up of more than one element, so there is special syntax for array declaration.

Example:

```
String[] cars;
```

Initializing Arrays

```
public static void main(String[] args) {  
    int[] arr1 = new int[4];  
    arr1[0] = 10;  
    arr1[1] = 20;  
    arr1[2] = 30;  
    arr1[3] = 40;  
  
    int[] arr2 = {10, 20, 30, 40};  
}
```

Accessing and Updating Array Elements

Accessing and updating array elements is similar between Java and other languages.

Example:

```
int[] arr3 = {10, 20, 30, 40};  
int x = arr3[2]; // 30  
arr3[0] = 5; // arr3 is now [5, 20, 30, 40]
```

Iterating Through Arrays

Traditional for Loop

```
for (int i = 0; i < arr1.length; i++) {  
    System.out.println(arr1[i]);  
}
```

Enhanced for Loop (similar to for-in)

```
for (int element : arr1) {  
    System.out.println(element);  
}
```

Two-Dimensional Arrays

```
public static void main(String[] args) {  
    String[][] multiArr = {  
        {"a", "b", "c"},  
        {"1", "2", "3"},  
        {"i", "ii", "iii"}  
    };  
  
    for (int i = 0; i < multiArr.length; i++) {  
        for (int j = 0; j < multiArr[i].length; j++) {  
            System.out.println( multiArr[i][j] );  
        }  
    }  
}
```



Recap

Java and other languages have many similarities (C-based, etc.), but there are important differences.



Static vs dynamic typing and variable declaration



Truth and truthiness



Comparison (triple equals)



Functions and methods



Tools



High vs. low ceremony



Jira and agile