Exploring Java Programming After Java 8

SIGCSE 2024 – Portland, OR

March 2024

TEXAS A&M UNIVERSITY

Shawn Lupoli

Dept. of Computer Science and Engineering

ORACLE
Java

Heather Stephens

Senior Director Java Product Management

Jamila Cocchiola

Product Manager – 6-12 Curriculum

ORACLE
Java

Paul Sandoz

Java Library Architect

ORACLE
Java

Crystal Furman

Director Oracle Java in Education

Agenda

1. State of Java
2. Hands-on Java 21
3. Computer Vision
4. Stay Connected

Please take breaks as needed! ☺

Code files can be found here: https://github.com/clfurman/SIGCSECodeFiles

Java™

29 years of innovation

Java continues to be an **important choice** for enterprises and developer

The Java technology **innovation pipeline** has never been richer

Java 22 available on March 19

State of Java

Java™

Moving Java forward

- Trust

  Continue delivering an open and transparent development model.

- Innovation

  Thoughtfully invest time to understand and evaluate where the modern application development landscape is going.

- Predictability

  Evolve Java predictably and incrementally, which offers efficient, stable and secure innovation.

#1
Language for today's Technology Trends

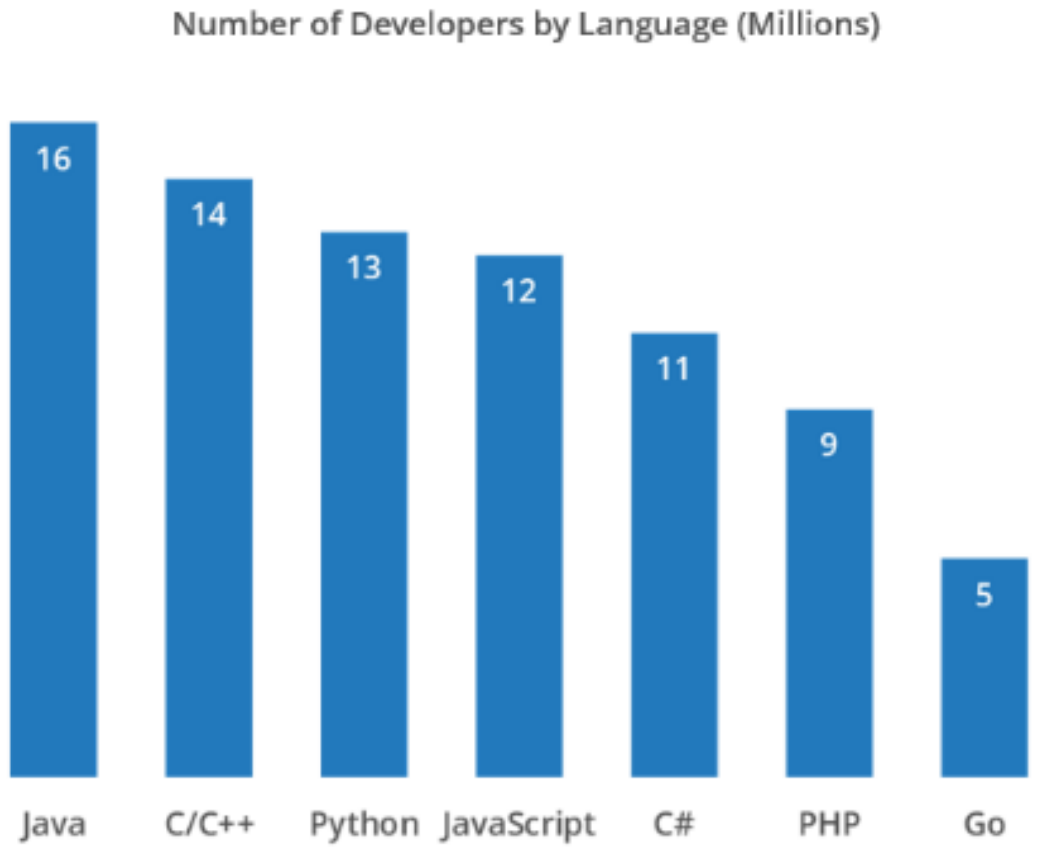
#1
Overall Enterprise/IT Organizational Use

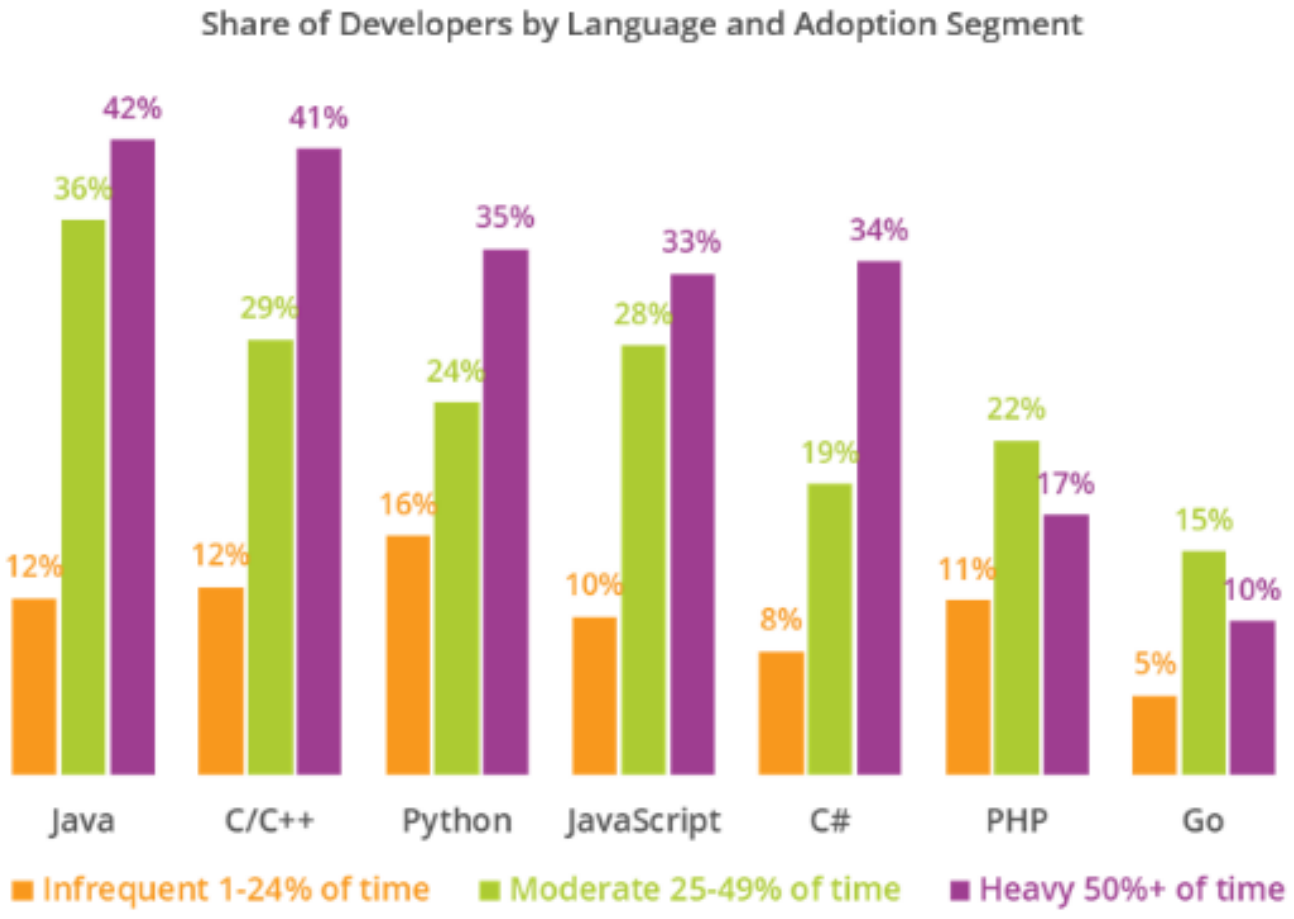
63B
Active JVMs

41B
Cloud-based JVMs

* VDC 2023 study

~16M million developers use Java out of a total full-time developer population of 17.5 million*





*Source: IDC Developer View 2023: Worldwide Survey Findings, Doc # US51065023

Choosing what to work on

JDK expectations for compatibility, stability, security, and performance are extreme

- Evolving use-cases
- Improvements in hardware
- New programming paradigms
- New trends or hardware architectures
- Endeavor to understand problems
- Look to identify synergies
- Avoid attempting to shoehorn solutions
- Avoid the temptation to shortcut

THOUGHTFUL EVOLUTION

"Tip and Tail"

Conservatism

Compatibility

Don't alienate users

Innovation

Adapting to change

Fixing mistakes

Java™

Select projects from OpenJDK – openjdk.org

Project	Summary	Pain point	"Obvious" Competition
Loom	Lightweight concurrency	"Threads are too expensive, don't scale"	Go, Elixir
ZGC	Sub-millisecond GC pauses	"GC pauses are too long"	C, Rust
Panama	Native code and memory interop SIMD Vector support	"Using native libraries is too hard" "Numeric loops are too slow"	Python, C
Amber	Right-sizing language ceremony	"Java is too verbose" "Java is hard to teach"	C#, Kotlin
Leyden	Faster startup and warmup	"Java starts up too slowly"	Go
Valhalla	Value types and specialized generics	"Cache misses are too expensive" "Generics and primitives don't mix"	C, C#
Babylon	Foreign programming model interop	"Using GPUs is too hard"	LinQ, Julia

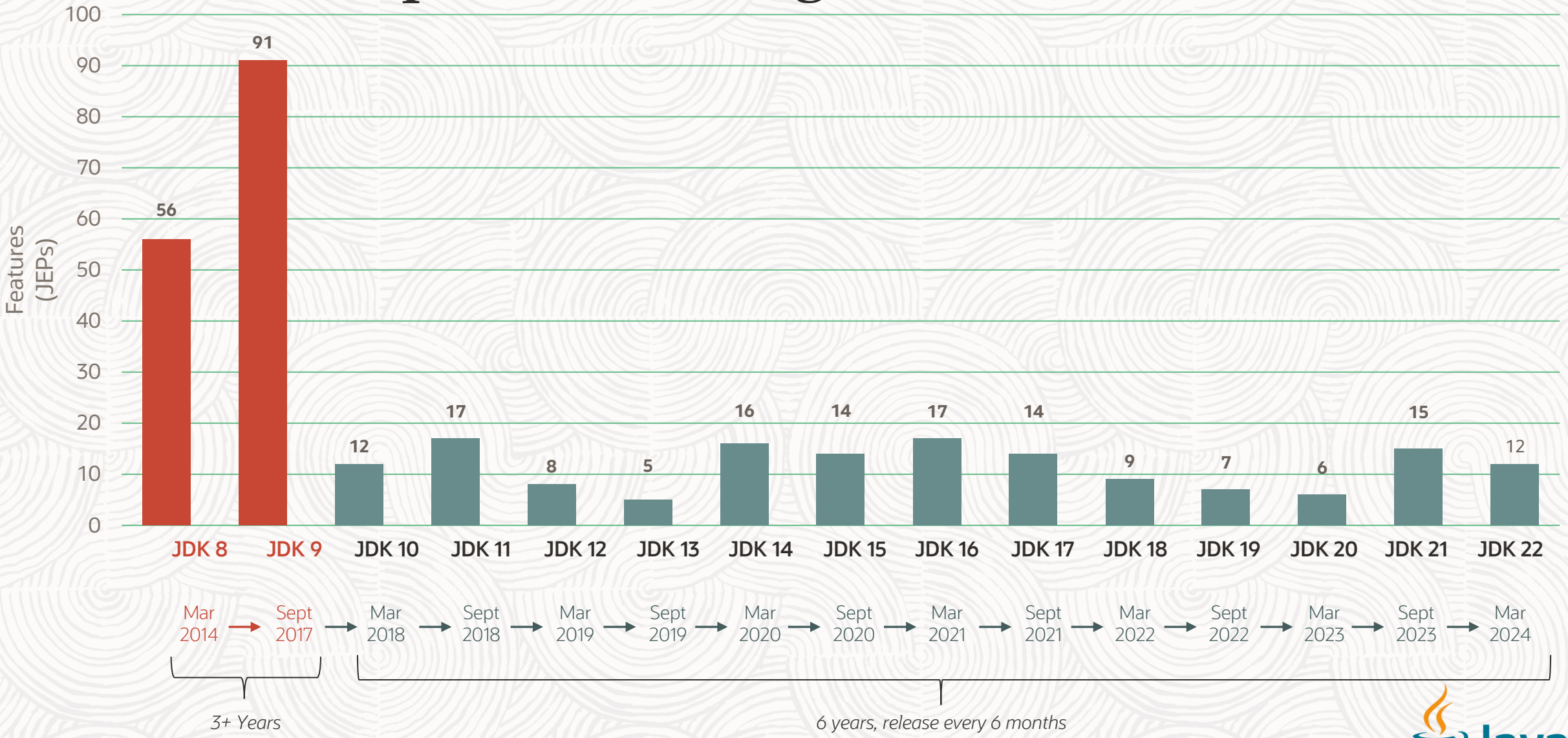
The JDK Enhancement Process

- OpenJDK Projects have features.

- Potential new features are proposed openly in the community via a document called a JDK Enhancement Proposal or JEP.

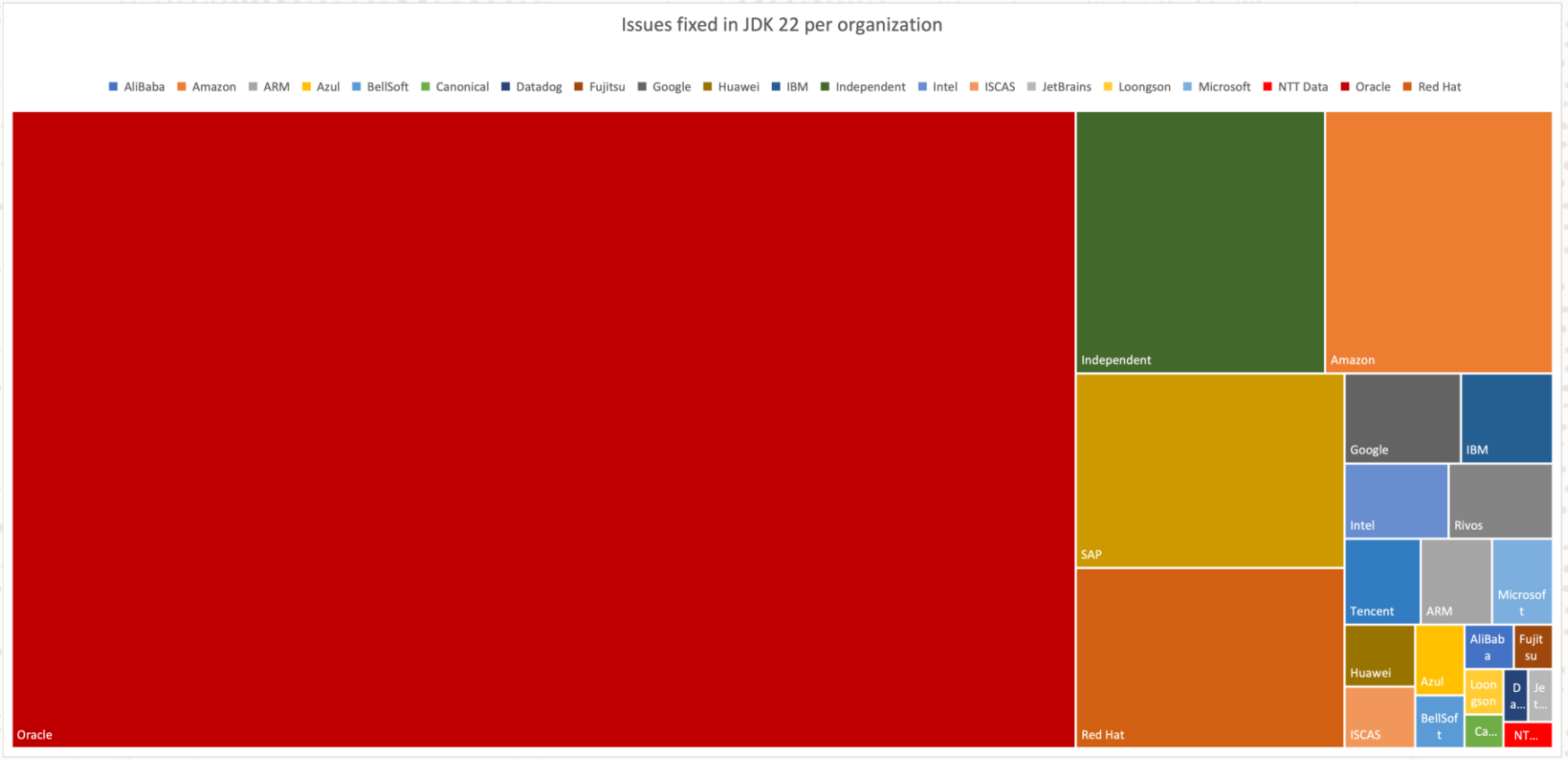
- Each JEP goes through a lifecycle.
  - Each must be approved for inclusion with a target release.
  - Each goes through at least 1 preview cycle to address feedback. Most go through 2-3 previews but it could be more.
  - Once feedback is addressed and the community approves the implementation, the JEP is targeted for a release as a final candidate.
  - With each release revision, the proposal documentation is updated to reflect the current state of the proposal and it is given a new identifying number to ensure traceability through history on versions of the idea.

How does OpenJDK manage releases?



Led by Oracle

- Leading author & contributor of Java technology.
- Leading sponsor & steward of the Java ecosystem.
- Leading driver of platform innovation.



JDK 21 Features

Language and tooling

- Record Patterns [440]
- Pattern Matching for switch [441]
- String Templates Preview [430]
- Unnamed Patterns and Variables Preview [443]
- Unnamed Classes and Instance Main Methods Preview [445]

Performance and Runtime

- Generational ZGC [439]
- Virtual Threads [444]

Libraries

- Sequenced Collections [431]
- Key Encapsulation Mechanism API [452]
- Foreign Function & Memory API Preview [442]
- Vector API Incubator [448]
- Structured Concurrency Preview [453]
- Scoped Values Preview [446]

Stewardship and housekeeping

- Prepare to Disallow the Dynamic Loading of Agents [451]
- Deprecate the Windows 32-bit x86 Port for Removal [449]

Java™

JDK features since JDK 17

Language and tooling
- Record Patterns [440]
- Pattern Matching for switch [441]
- String Templates Preview [430]
- Unnamed Patterns and Variables Preview [443]
- Unnamed Classes and Instance Main Methods Preview [445]
- **Code snippets in API documentation [413]**

Libraries
- Sequenced Collections [431]
- Key Encapsulation Mechanism API [452]
- Foreign Function & Memory API Preview [442]
- Vector API Incubator [448]
- Structured Concurrency Preview [453]
- Scoped Values Preview [446]
- **Simple web server [408]**
- **Internet Address Resolution SPI [418]**

Performance and Runtime
- Generational ZGC [439]
- Virtual Threads [444]
- **Linux/RiscV Port [422]**

Stewardship and housekeeping
- Prepare to Disallow the Dynamic Loading of Agents [451]
- Deprecate the Windows 32-bit x86 Port for Removal [449]
- **UTF-8 by default [400]**
- **Deprecate finalization for removal [421]**
- **Reimplement reflection [416]**

JDK features since JDK 11

Language and tooling
- Record Patterns [440]
- Pattern Matching for switch [441]
- String Templates Preview [430]
- Unnamed Patterns and Variables Preview [443]
- Unnamed Classes and Instance Main Methods Preview [445]
- **Code snippets in API documentation [413]**
- **Restore always-strict floating point semantics [306]**
- **Sealed classes [409]**
- **Pattern Matching for instanceof [394]**
- **Records [395]**
- **Text Blocks [378]**
- **Switch Expressions [361]**
- **Packaging tool [392]**

Performance and Runtime
- Generational ZGC [439]
- Virtual Threads [444]
- **Linux/RiscV Port [422]**
- **New MacOS rendering pipeline [382]**
- **AArch64 port for MacOS, Windows [391, 388]**
- **ZGC and improvements [377, 376, 351]**
- **Alpine Linux Port [386]**
- **Elastic metaspace [387]**
- **Hidden classes [371]**
- **Shenandoah Garbage Collector [379]**
- **G1 Improvements [345, 344, 346]**
- **JFR Event Streaming [349]**
- **Helpful NullPointerException [358]**
- **Dynamic and Default CDS Archives [350, 341]**

Libraries
- Sequenced Collections [431]
- Key Encapsulation Mechanism API [452]
- Foreign Function & Memory API Preview [442]
- Vector API Incubator [448]
- Structured Concurrency Preview [453]
- Scoped Values Preview [446]
- **Simple web server [408]**
- **Internet Address Resolution SPI [418]**
- **Enhanced pseudo-random number generators [356]**
- **Context-specific deserialization filters [415]**
- **Unix domain socket channels [380]**
- **Edwards Curve Digital Signatures [339]**
- **Non-Volatile Mapped ByteBuffer [352]**
- **JVM Constants API [334]**

Stewardship and housekeeping
- Prepare to Disallow the Dynamic Loading of Agents [451]
- Deprecate the Windows 32-bit x86 Port for Removal [449]
- **UTF-8 by default [400]**
- **Deprecate finalization for removal [421]**
- **Reimplement reflection [416]**
- **Deprecate the Applet API for removal [398]**
- **Strongly encapsulate JDK internals [403]**
- **Deprecate and remove RMI Activation [385, 407]**
- **Deprecate the security manager for removal [411]**
- **Migrate to Git / GitHub [357, 369]**
- **Reimplement reflection [416]**
- **Remove Nashorn [372]**
- **Reimplement DatagramSocket [373]**
- **Reimplement Socket [353]**
- **Disable and Deprecate Biased Locking [374]**
- **Deprecate and remove Solaris and Sparc Ports [362, 381]**
- **Remove ConcurrentMarkSweep collector [363]**
- **Remove Pack200 [367]**

Java™

JDK 22

12 JDK Enhancement Proposals (JEPs) delivered

Plus thousands of performance, stability and security updates

Project Amber

- 447: Statements before super (...) **Preview**
- 456: Unnamed Variables & Patterns
- 459: String Templates **Second Preview**
- 463: Implicitly Declared Classes and Instance Main Methods **Second Preview**

Project Loom

- 462: Structured Concurrency **Second Preview**
- 464: Scoped Values **Second Preview**

Core Libraries and Tools

- 457: Class-File API **Preview**
- 458: Launch Multi-File Source-Code Programs
- 461: Stream Gatherers **Preview**

Project Panama

- 454: Foreign Function & Memory API
- 460: Vector API **Seventh Incubator**

Performance Updates

- 423: Regional Pinning for G1

openjdk.org/projects/jdk/22

Paving the On Ramp

Before

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

- Classes for modeling and organization
- Static vs instance behavior
- Command line interaction, arrays
- Magic method name
- Access control
- Static fields

After

```
void main() {
    println("Hello!");
}
```

dev.java

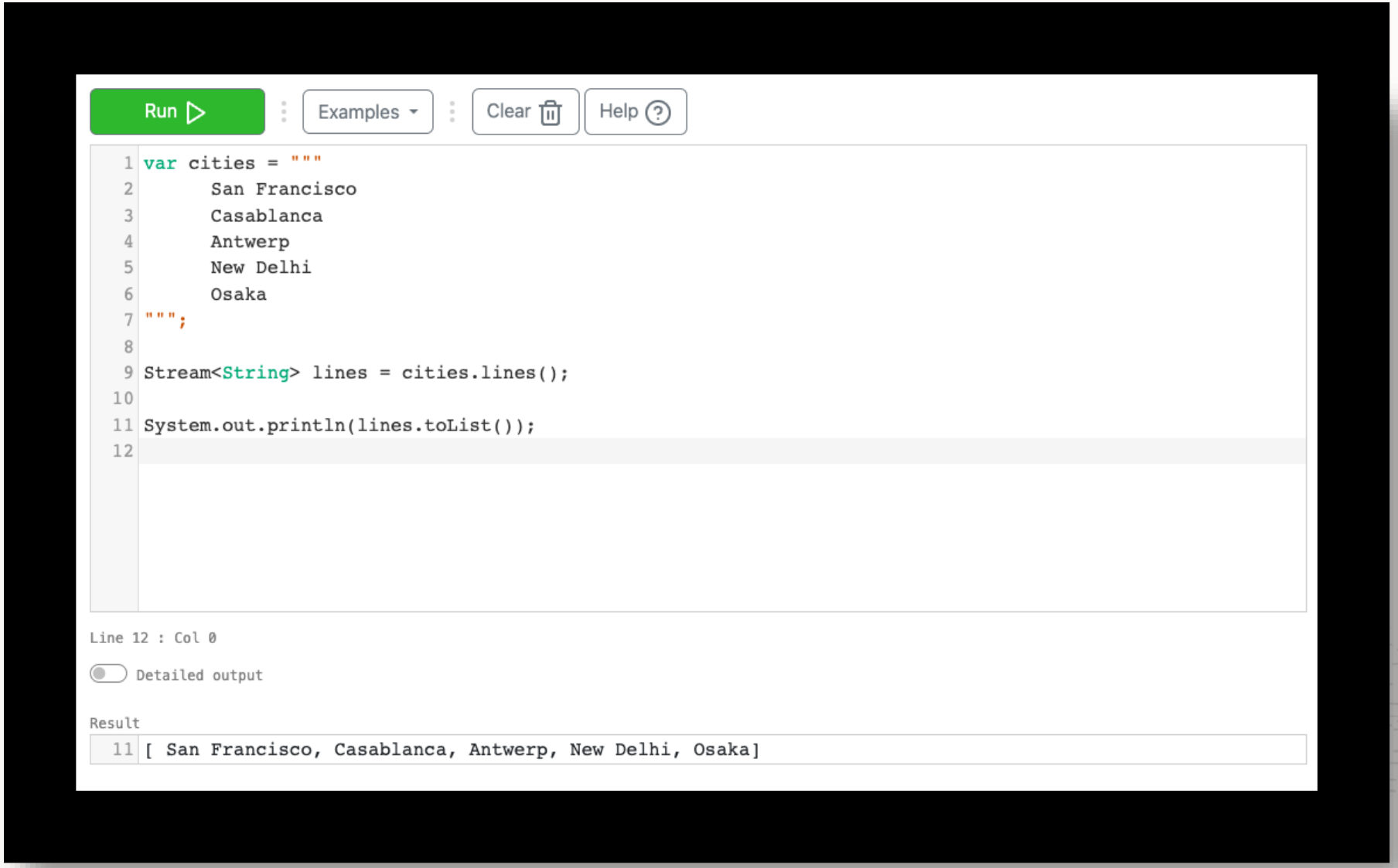
Think of dev.java as java.com

Your source for everything Java!!

Java plug-in for VS Code

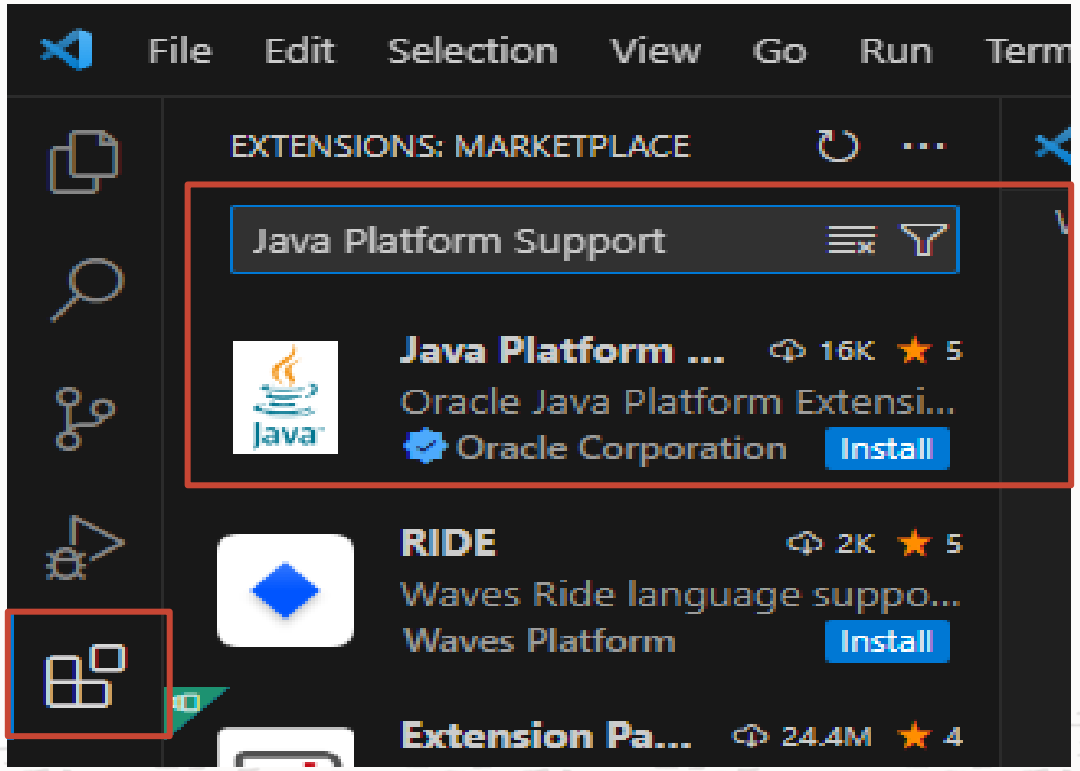
- Java Platform plugin availability in the VS Code marketplace
- Exists with other Oracle plugins in the VS Code marketplace
- Oracle's VS Code extension for Java is based on the OpenJDK's `javac` compiler for code editing and compilation and on the OpenJDK's debugger interface for debugging
- Allows Oracle to offer VS Code IDE support for new JDK features as soon as they are introduced, even during Early Access of the JDK
- Oracle’s VS Code extension supports the current

Java Playground at Dev.java



Visual Studio Code Set Up

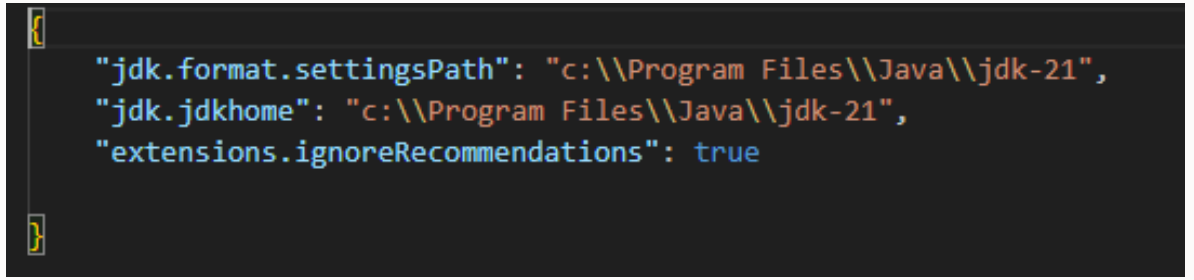
1. Download Visual Studio Code here: https://code.visualstudio.com/
2. Install *Java Platform Support* by clicking on *Extensions* and searching *Java Platform Support*.



Visual Studio Code Set Up

1. Download Visual Studio Code here: https://code.visualstudio.com/
2. Install *Java Platform Support* by clicking on *Extensions* and searching *Java Platform Support*.
3. Go to View … Command Palette and Select *Preferences: Open User Settings (JSON)* from the dropdown. Be sure to add "jdk.jdkhome": add then the path to where you installed the Java JDK. This is most likely in the Program Files if you have a windows set up.

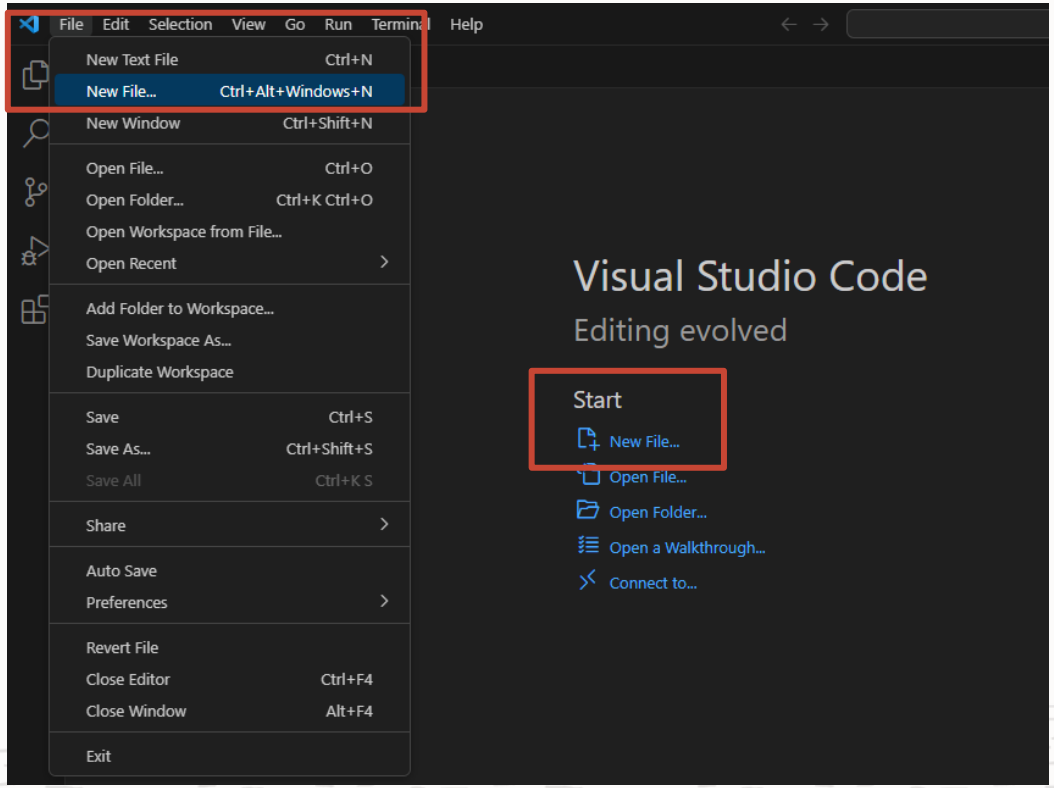
NOTE: The path might use a single back-slash, such as c:\Program Files\Java\jdk-21 In this case, you will need to add an additional back-slash for each as shown in the example below.



Implicitly Declared Classes & Instance Main Methods with VS Code

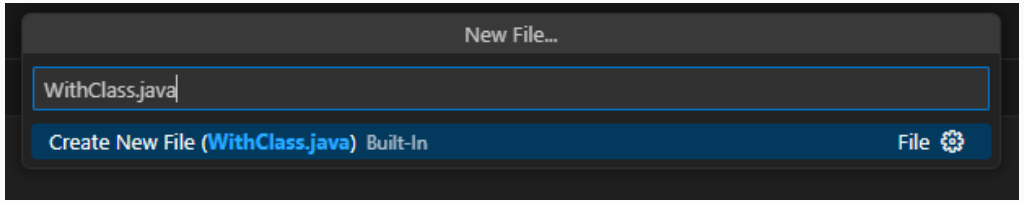
Use for fast prototyping code.

1. In Visual Studio, create a new file.



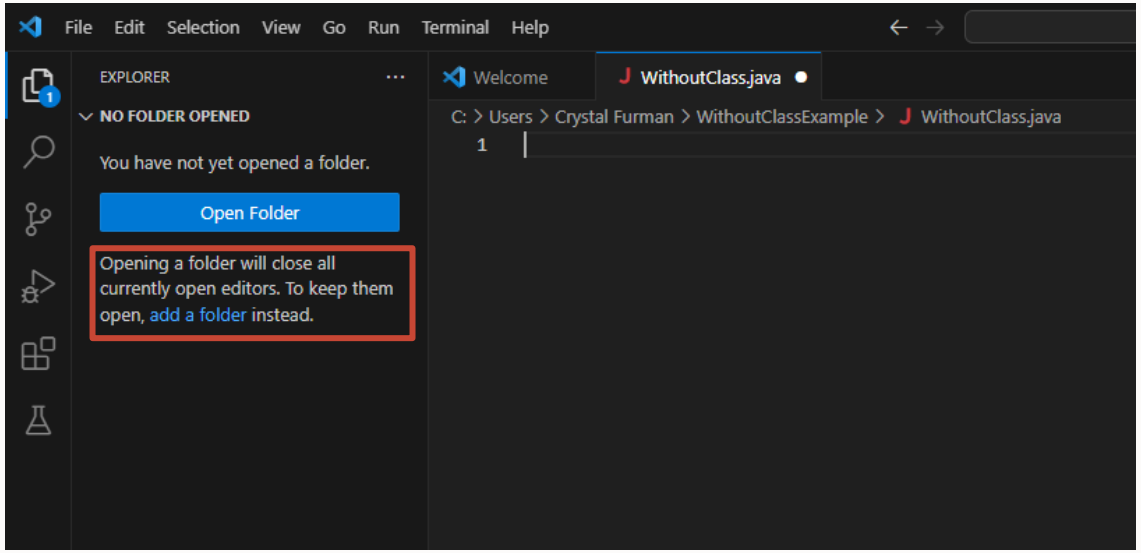
Implicitly Declared Classes & Instance Main Methods with VS Code

1. In Visual Studio, create a new file.
2. Name the file with the .java extension.



Implicitly Declared Classes & Instance Main Methods with VS Code

1. In Visual Studio, create a new file.
2. Name the file with the .java extension.
3. When directed to save the file, select or create a new folder for your files.
4. Even though you saved in the folder, you will still need to open the folder or add a folder. When you click on add a folder, navigate to the folder you saved the .java file into (it should be automatically selected) and click “add”.



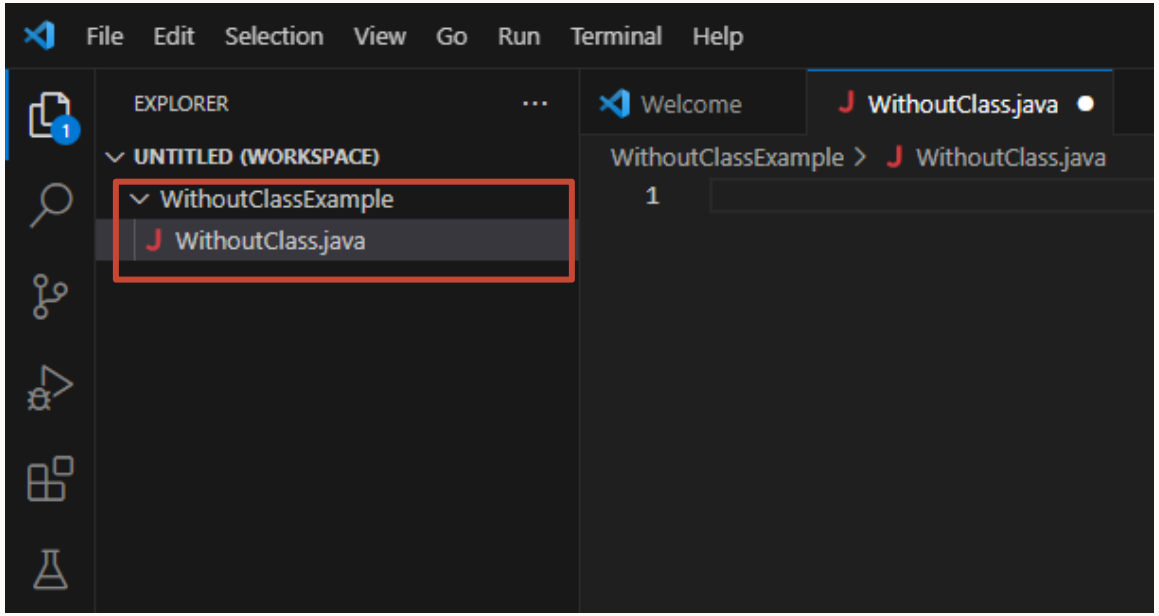
NOTE: You might be asked if you trust the authors of the files in the folder. Click “yes”.

Implicitly Declared Classes & Instance Main Methods with VS Code

1. In Visual Studio, create a new file.
2. Name the file with the .java extension.
3. When directed to save the file, select or create a new folder for your files.
4. Even though you saved in the folder, you will still need to open the folder or add a folder. When you click on add a folder, navigate to the folder you saved the .java file into (it should be automatically selected) and click “add”.

NOTE: You might be asked if you trust the authors of the files in the folder. Click “yes”.

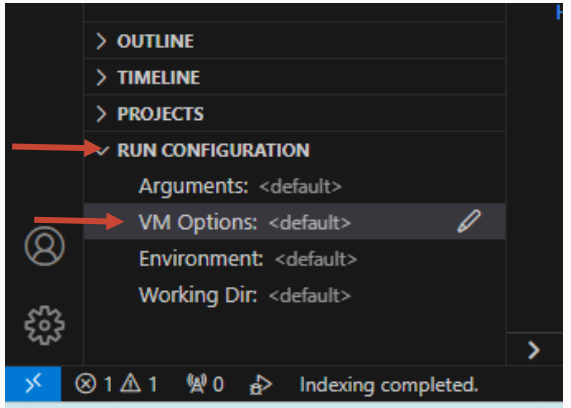
5. You will see the .java file is now associated with the folder.

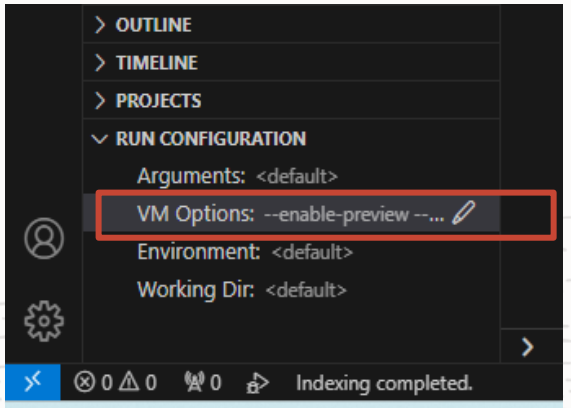


Implicitly Declared Classes & Instance Main Methods with VS Code

6. Open Run Configurations and edit VM Options to add

```
--enable-preview --source 21
```





Implicitly Declared Classes & Instance Main Methods with VS Code

6. Open Run Configurations and edit VM Options to add

```
--enable-preview --source 21
```

7. Add a main method to your class as follows:

```
void main(){
  System.out.println ("Hi there class!");
}
```

8. You will notice two new commands above the main method: Run main | Debug main. Click on Run main and your output will show in the terminal window.

Implicitly Declared Classes & Instance Main Methods with VS Code

6. Open Run Configurations and edit VM Options to add

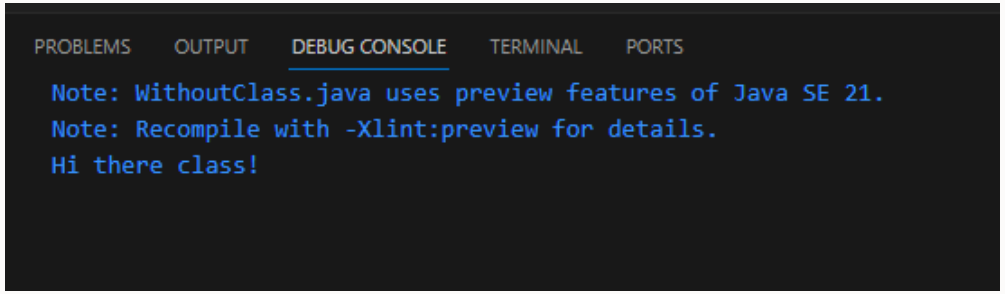
```
--enable-preview --source 21
```

7. Add a main method to your class as follows:

```
void main(){
 System.out.println ("Hi there class!");
}
```

8. You will notice two new commands above the main method: Run main | Debug main. Click on Run main and your output will show in the terminal window.

Terminal Window, in the DEBUG CONSOLE tab:



5 Minutes

Discussion on Use in CS Course of Study

1. How might you incorporate Implicitly Declared Classes and Instance Main Methods as an instructional tool?
2. This is a preview feature, do you have any initial feedback?

Java Playground

The Java Playground is a simple online tool that helps you to explore Java Language features. No setup required! The Java Playground currently runs on the Oracle JDK – Java 21 with Preview Features enabled.

1. Go to dev.java
2. Click on “Playground”
3. Type some code and Run
   - `System.out.println ("Hello World");`
   - An arithmetic expression to demonstrate order of operations (e.g., `SOP (5 % 2);`)
   - Demonstrate foundational content, such as a simple if statement or loop
4. Explore the various pre-loaded program segments in Examples

5

Discussion on Use in CS Course of Study

1. How might you use Java Playground as an instructional tool?
   - Lab?
   - Tutorial?
   - Examples?
   - Other?

String Templates

JEP 459 | 2nd Preview Java 22

Java™

String Templates

- Allows a more easily embedding of expressions into `String` content.
- A safer way to accomplish string interpolation.
- Use the string template `STR` and the escape sequence `\{` prior to the `expression` and `}` at the end of the expression to embed expressions into the strings.

String Templates – In Java Playground

1. Go to dev.java
2. Click on “Playground”

```
int x = 5;
int y = 10;
String concatS = "The sum of " + x + " and " + y +
                 " is " + (x + y);

String s =  STR."The sum of \{x} and \{y} is \{x + y}";
```

Inserts the value of `x`

Inserts the value of `y`

Inserts the value of `x + y`

3. Enter the code into the Java Playground, include output statements for `concatS` and `s` then click Run.

5 Minutes

Discussion on String Templates in CS

1. Where does this feature fit into a CS course of study? CS1? CS2?
2. At what point in the course would you want to include String Templates?
3. What might be moved up or out to make room for this new content?

Records

JEP 395 | Java 16

Record

- Records are better when we are representing data.
- They allow us to treat data as data.
- Record data is immutable.
- Records are used in a Data Oriented Programming approach vs. an Object Oriented Programming approach.
- They allow us to create a new data type to model data.
- While we can write our own methods to perform operations on the data, we aren't writing methods as "behaviors".
- Records provide constructors, private instance variables, and accessor methods automatically.

Point Class vs Point Record Comparison

```java
public class Point {

    private final int x;
    private final int y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(int myX, int myY) {
        x = myX;
        y = myY;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```java
record PointRecord(int x, int y) {

}
```


Java™

Point Class vs Point Record Comparison

```java
public class Point {

    private final int x;
    private final int y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(int myX, int myY) {
        x = myX;
        y = myY;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```java
record PointRecord(int x, int y) {

}
```

- Place all the instance variables needed in the parameter list to create your private instance variables.

Java™

Point Class vs Point Record Comparison

```
public class Point {

    private final int x;
    private final int y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(int myX, int myY) {
        x = myX;
        y = myY;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```
record PointRecord(int x, int y) {

}
```

- Creates the constructor for the record class.
- NOTE: A default constructor is not provided.

Java™

Point Class vs Point Record Comparison

```java
public class Point {

    private final int x;
    private final int y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(int myX, int myY) {
        x = myX;
        y = myY;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```java
record PointRecord(int x, int y) {

}
```

- Accessor methods for each instance variable is created. The name of the method is the same name as the variable name. In this case:
  - `x()`
  - `y()`

Other Methods Automatically Included

- `toString()`
- `equals()`
- `hashCode()`

toString Method Included in Record

```
public static void main(String[] args) {
    System.out.println("Hello World!");
    PointRecord p1 = new PointRecord (3, 5);
    Point p2 = new Point (5, 7);


    System.out.println (p1);
    System.out.println (p2);
```

Output:

```
Hello World!
PointRecord[x=3, y=5]
org.SIGCSEWorkshop.RecordExample.Point@6e8cf4c6
```


Java™

equals Method included in Record

```
public static void main(String[] args) {
    System.out.println("Hello World!");
    PointRecord p1 = new PointRecord (3, 5);
    PointRecord p2 = new PointRecord (5, 7);
    PointRecord p3 = new PointRecord (5, 7);
    Point p4 = new Point (3, 5);
    Point p5 = new Point (5, 7);
    Point p6 = new Point (5, 7);

    System.out.println (p1.equals(p2));
    System.out.println (p2 == p3);
    System.out.println (p2.equals (p3));

    System.out.println (p4.equals (p5));
    System.out.println (p5 == p6);
    System.out.println (p5.equals(p6));
}
```

Output:

```
Hello World!
false
false
true
false
false
false
```

Adding Methods to Records

```
record PointRecord (int x, int y){

    public double slope (PointRecord other)
    {
        try {
            return ((y - other.y)/(x - other.x) );
        } catch (Exception e) {
            System.out.println ("No Slope");
        }
        return -9999999;
    }
}
```


Java™

Adding Methods to Records

```
public static void main(String[] args) {
    System.out.println("Hello World!");
    PointRecord p1 = new PointRecord (3, 5);
    PointRecord p2 = new PointRecord (5, 7);
    PointRecord p3 = new PointRecord (5, 8);

    System.out.println (p1.slope(p2));
    System.out.println (p2.slope (p3));

}
```

Output:

```
Hello World!
1.0
No Slope
-9999999.0
```

Applications of Records – Meteorite Data

1. Navigate to GitHub here: https://github.com/clfurman/MeteoriteDataProjectFinal
2. Create Record classes to store the immutable data of each row in the data file. In this case, I chose to create two records:
   a) `Meteor`
   b) `GeoLocation`
3. Use the runner class to import the data and add the meteors with a mass less than 5000 to a new list.

10 Minutes

Java™

Discussion on Records in CS Courses

1. Where does this feature fit into a CS course of study? CS1? CS2?
2. At what point in the course would you want to include Records?
3. What might be moved up or out to make room for this new content?

Extending Meteorite Data Project

1. Go to... https://padlet.com/crystal_furman/Meteorite_Brainstorm

2. How can we extend the Meteorite Data Project?

Sealed Classes and Interfaces

JEP 409 | Java 17

Sealed Interfaces

- Sealed classes and interfaces restrict which other classes or interfaces can extend or implement them
- Allows the originator to control which code is responsible for implementing it.
- Supports pattern matching with switch, allowing switch cases to be exhaustive without the need for a default clause.

Keyword for creating a sealed interface

Keyword used to identify with class or interface names

```
sealed interface Animal permits Dog, Cat {

}
```

The classes or interfaces that are allowed to extend or implement `Animal`

Application of Sealed Interfaces – Vet Office

1. Go to https://github.com/clfurman/PatternMatchingSealedFinal
2. Complete the first three modification to create the following:
   - `Dog` Record
   - `Cat` Record
   - `LicenseName` Record
   - Sealed `Animal` Interface
3. NOTE: All four of these files can be placed in the same .java file. The name of this file needs to be the same name as the class that contains the main method.

10 Minutes

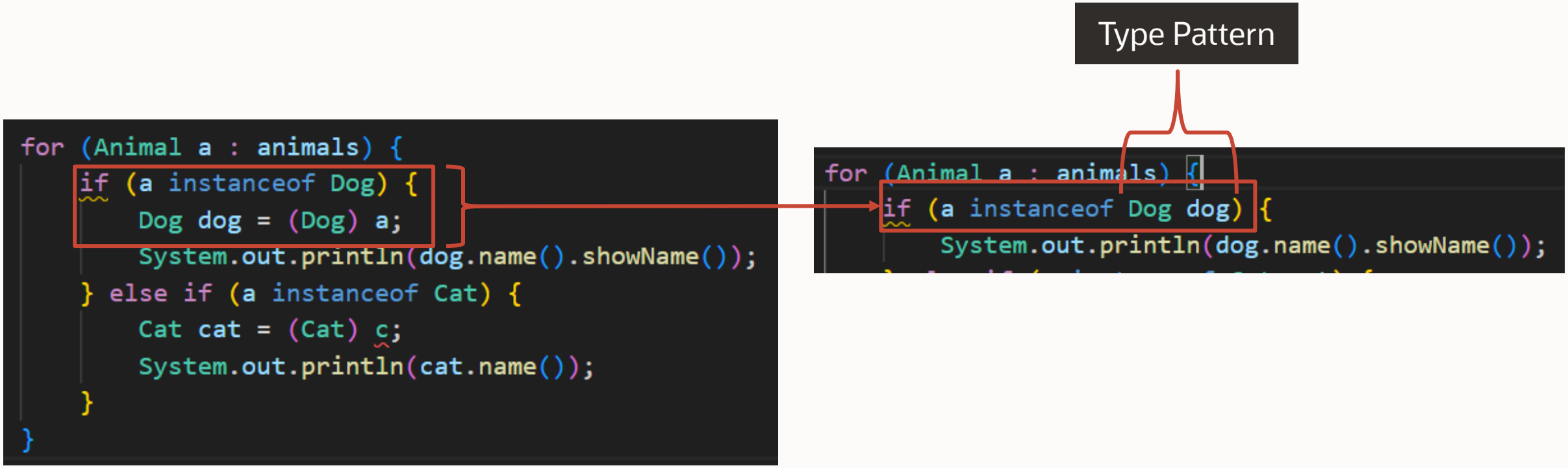
Record Patterns

JEP 440 | Java 21

Type Patterns

Combines the following into one statement:

- Type checking
- Variable declaration
- Cast / assignment



Application of Pattern Matching – Vet Office

1. Go to https://github.com/clfurman/PatternMatchingSealedFinal
2. Complete the fourth modification to print the name of each animal in the list.
3. NOTE: All four of these files can be placed in the same .java file. The name of this file needs to be the same name as the class that contains the main method.

10 Minutes

Pattern Matching for switch

JEP 441 | Java 21

Pattern Matching for switch

- The line `case Dog dog:` uses pattern matching to declare `dog` of type `Dog` and assign it to `(Dog)a`, if `a` is a type of Dog.
- When using `switch` statements, typically a default case is required to make the `switch` exhaustive.
- When `sealed` types are used, `default` isn't required since we have a strict set of cases and therefore the cases are exhaustive.

```
for (Animal a : animals) {
    switch (a) {
        case Dog dog: {
            System.out.println(dog.name().showName());
            break;
        }
        case Cat cat: {
            System.out.println(cat.name());
            break;
        }
    }
}
```


Java™

Application of Pattern Matching – Vet Office

1. Go to https://github.com/clfurman/PatternMatchingSealedFinal
2. Complete the modification #5 and 6 to update the if statement to use a switch statement instead.
3. Have more time? Complete the extension!

5 Minutes

Discussion on Pattern Matching in CS Courses

1. Where does this feature fit into a CS course of study? CS1? CS2?
2. At what point in the course would you want to include Pattern Matching?
3. What might be moved up or out to make room for this new content?

Computer Vision

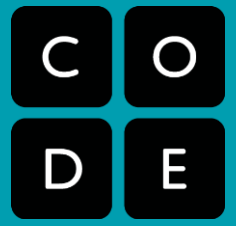
Code.org

Create a Code Studio account at:

studio.code.org

Join the section:

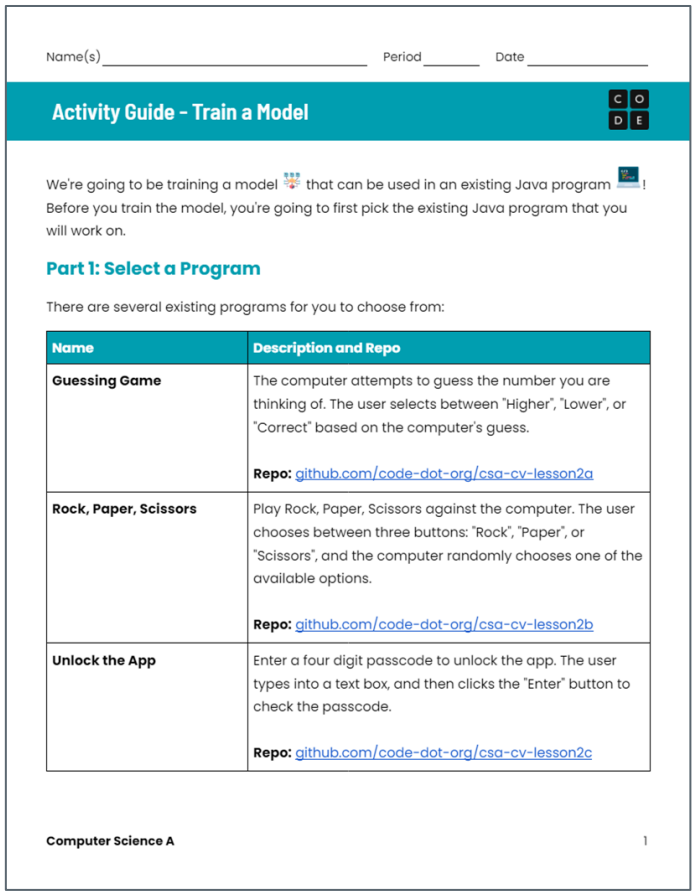
studio.code.org/join/SNGZDD



Chapter 2 - Lesson 6 Training Your Model

Software Engineering & Computer Vision

Warm Up



Name(s) _____ Period _____ Date _____

Activity Guide - Train a Model

We're going to be training a model that can be used in an existing Java program! Before you train the model, you're going to first pick the existing Java program that you will work on.

Part 1: Select a Program

There are several existing programs for you to choose from:

Name	Description and Repo
Guessing Game	The computer attempts to guess the number you are thinking of. The user selects between "Higher", "Lower", or "Correct" based on the computer's guess. **Repo:** github.com/code-dot-org/csa-cv-lesson2a
Rock, Paper, Scissors	Play Rock, Paper, Scissors against the computer. The user chooses between three buttons: "Rock", "Paper", or "Scissors", and the computer randomly chooses one of the available options. **Repo:** github.com/code-dot-org/csa-cv-lesson2b
Unlock the App	Enter a four digit passcode to unlock the app. The user types into a text box, and then clicks the "Enter" button to check the passcode. **Repo:** github.com/code-dot-org/csa-cv-lesson2c

Computer Science A 1

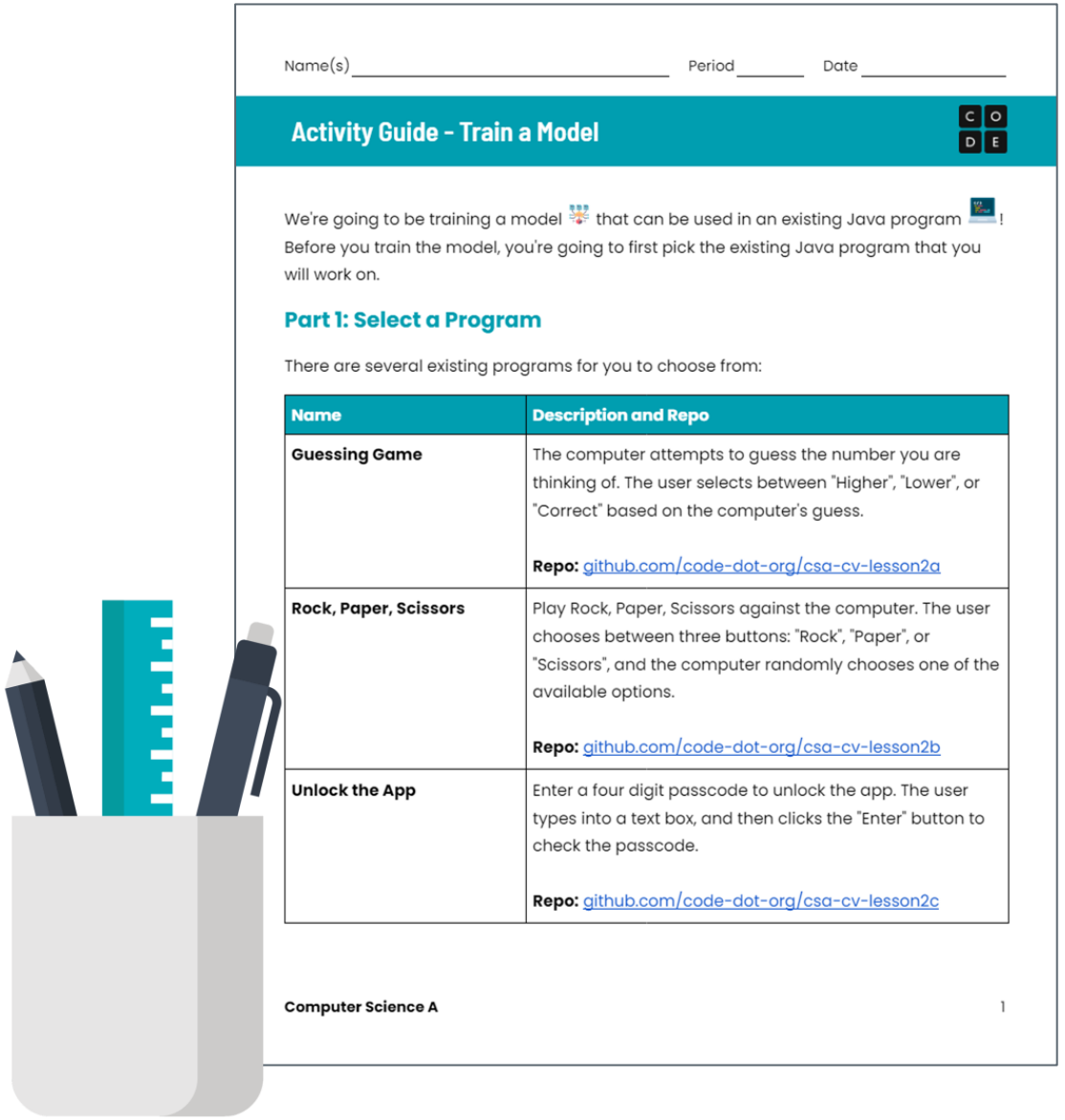
Train Your Model

You should have:

- ☑ Train Your Model activity guide

☑ Do This:

Complete the **Select a Program** portion of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Train a Model

We're going to be training a model that can be used in an existing Java program! Before you train the model, you're going to first pick the existing Java program that you will work on.

Part 1: Select a Program

There are several existing programs for you to choose from:

Name	Description and Repo
Guessing Game	The computer attempts to guess the number you are thinking of. The user selects between "Higher", "Lower", or "Correct" based on the computer's guess. **Repo:** github.com/code-dot-org/csa-cv-lesson2a
Rock, Paper, Scissors	Play Rock, Paper, Scissors against the computer. The user chooses between three buttons: "Rock", "Paper", or "Scissors", and the computer randomly chooses one of the available options. **Repo:** github.com/code-dot-org/csa-cv-lesson2b
Unlock the App	Enter a four digit passcode to unlock the app. The user types into a text box, and then clicks the "Enter" button to check the passcode. **Repo:** github.com/code-dot-org/csa-cv-lesson2c

Computer Science A 1

💡 Discuss:

How could you **incorporate an AI model** into your program?

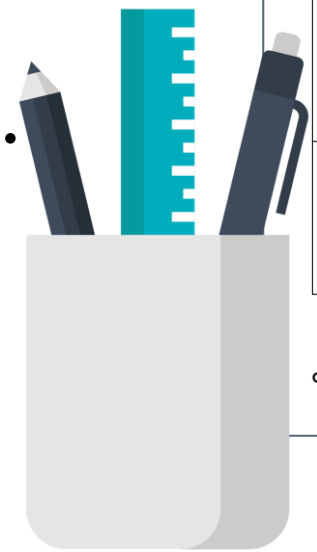
A **class** is a category that a machine learning model learns.

💡 Discuss:

What **classes** might you need for your model?

☑ Do This:

- ☒ Complete the **Teachable Machine** portion of the activity guide.
- ☒ Save your model as a **Tensorflow Savedmodel**.



Name(s) _____ Period _____ Date _____

Activity Guide - Train a Model

We're going to be training a model that can be used in an existing Java program !
Before you train the model, you're going to first pick the existing Java program that you will work on.

Part 1: Select a Program

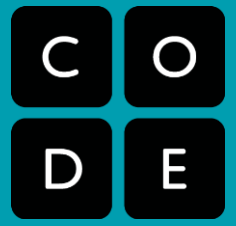
There are several existing programs for you to choose from:

Name	Description and Repo
Guessing Game	The computer attempts to guess the number you are thinking of. The user selects between "Higher", "Lower", or "Correct" based on the computer's guess. **Repo:** github.com/code-dot-org/csa-cv-lesson2a
Rock, Paper, Scissors	Play Rock, Paper, Scissors against the computer. The user chooses between three buttons: "Rock", "Paper", or "Scissors", and the computer randomly chooses one of the available options. **Repo:** github.com/code-dot-org/csa-cv-lesson2b
Unlock the App	Enter a four digit passcode to unlock the app. The user types into a text box, and then clicks the "Enter" button to check the passcode. **Repo:** github.com/code-dot-org/csa-cv-lesson2c

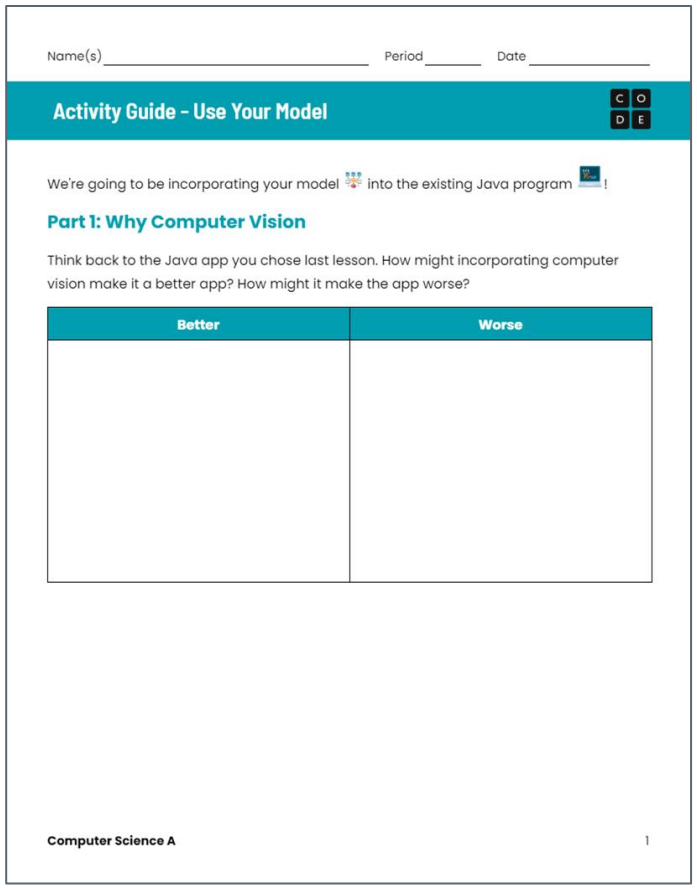
Discuss:

When did your model **not work**?

How did you **retrain** it?



Chapter 2 - Lesson 7
Using Your Model



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

Computer Science A 1

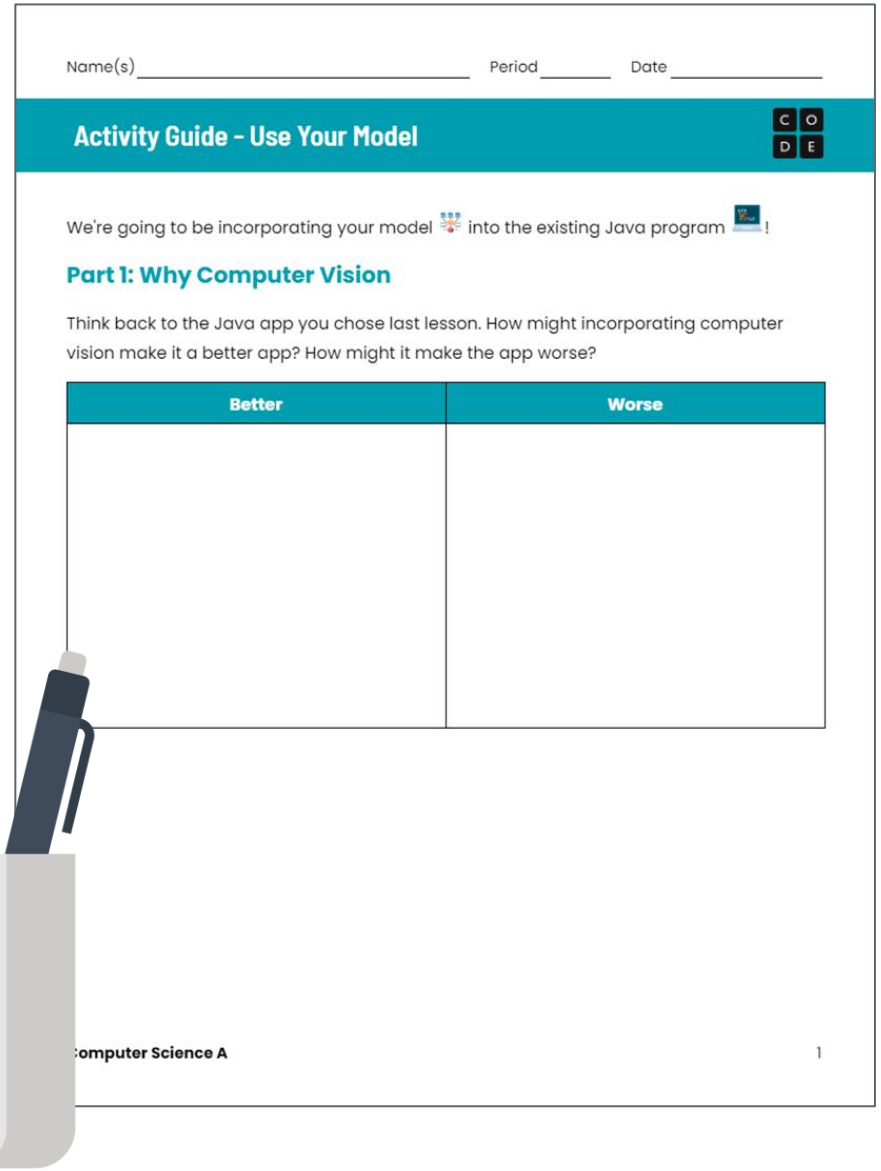
Use Your Model

You should have:

- ☑ Use Your Model activity guide

☑ Do This:

Complete the **Why Computer Vision** section of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

💡 Discuss:

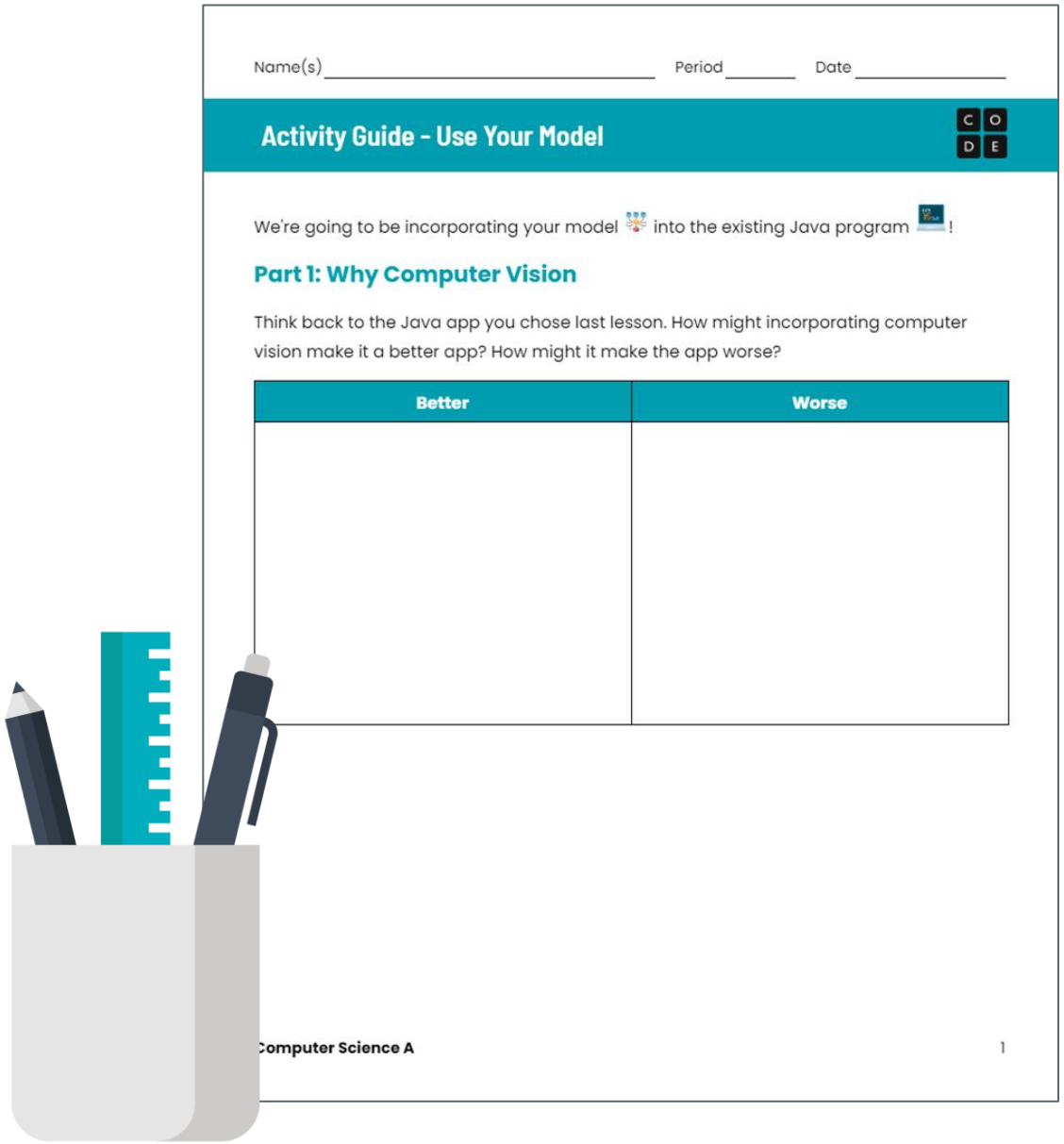
Think back to the Java app you chose last lesson.

How might **incorporating computer vision** make it a **better** app?

How might it make the app **worse**?

☑ Do This:

Work with your partner to complete the **Incorporate Your Model** section of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

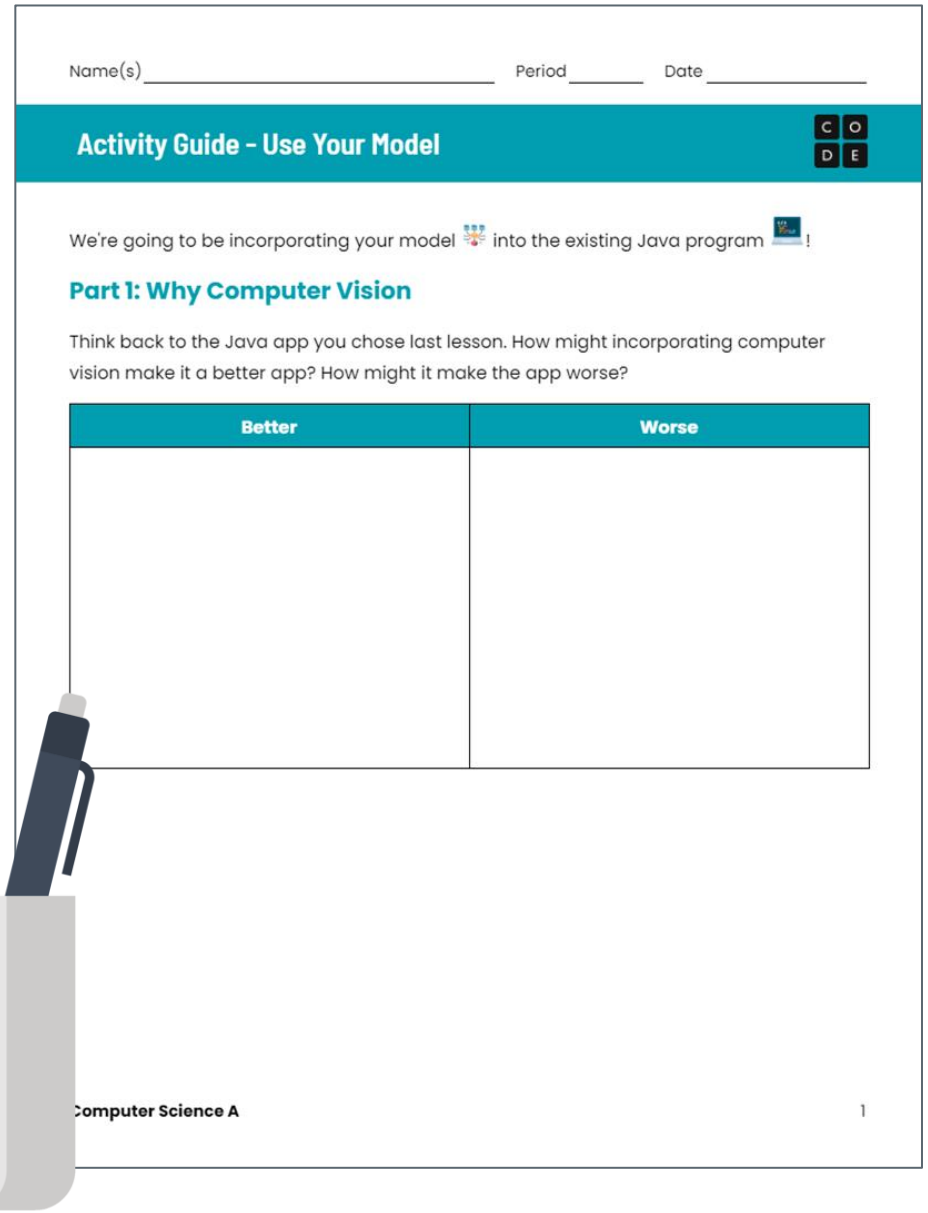
Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

Computer Science A 1

☑ Do This:

With your partner, complete the **Merge and Test Your App** section of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

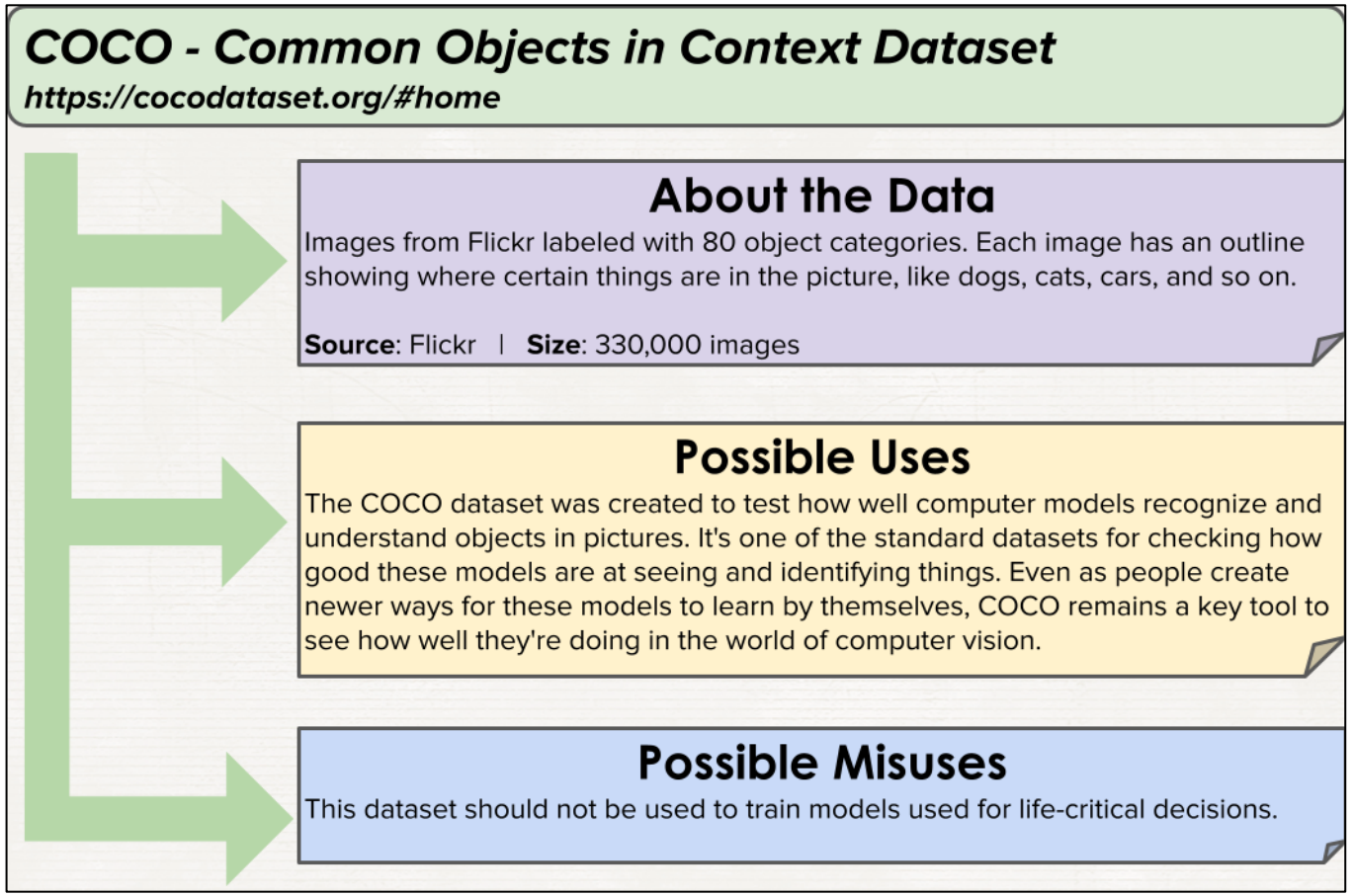
Part 1: Why Computer Vision

Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

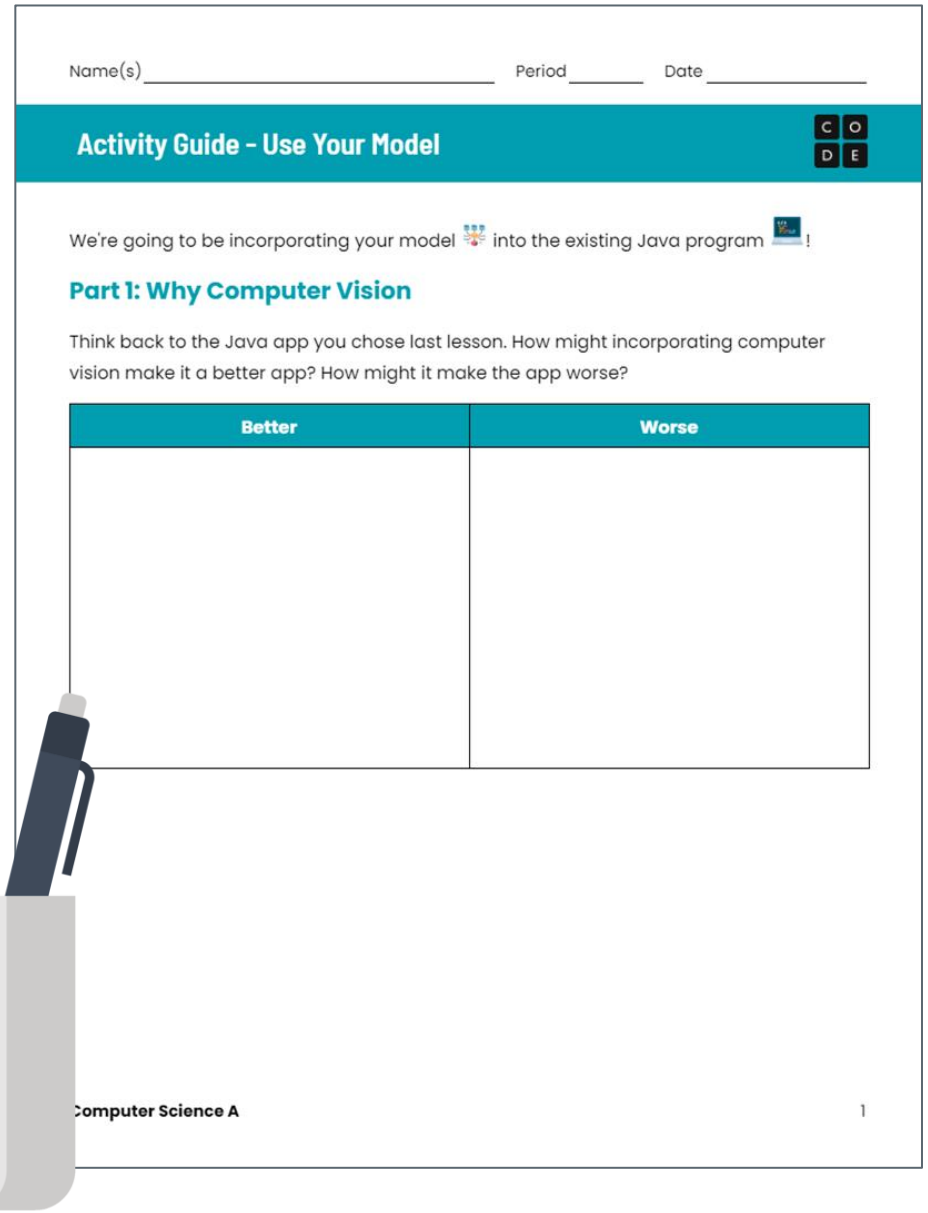
Computer Science A 1

Datasheets can be used to help **document the datasets** used to train our machine learning models.



☑ Do This:

With your partner, complete the **Create a Datasheet** section of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

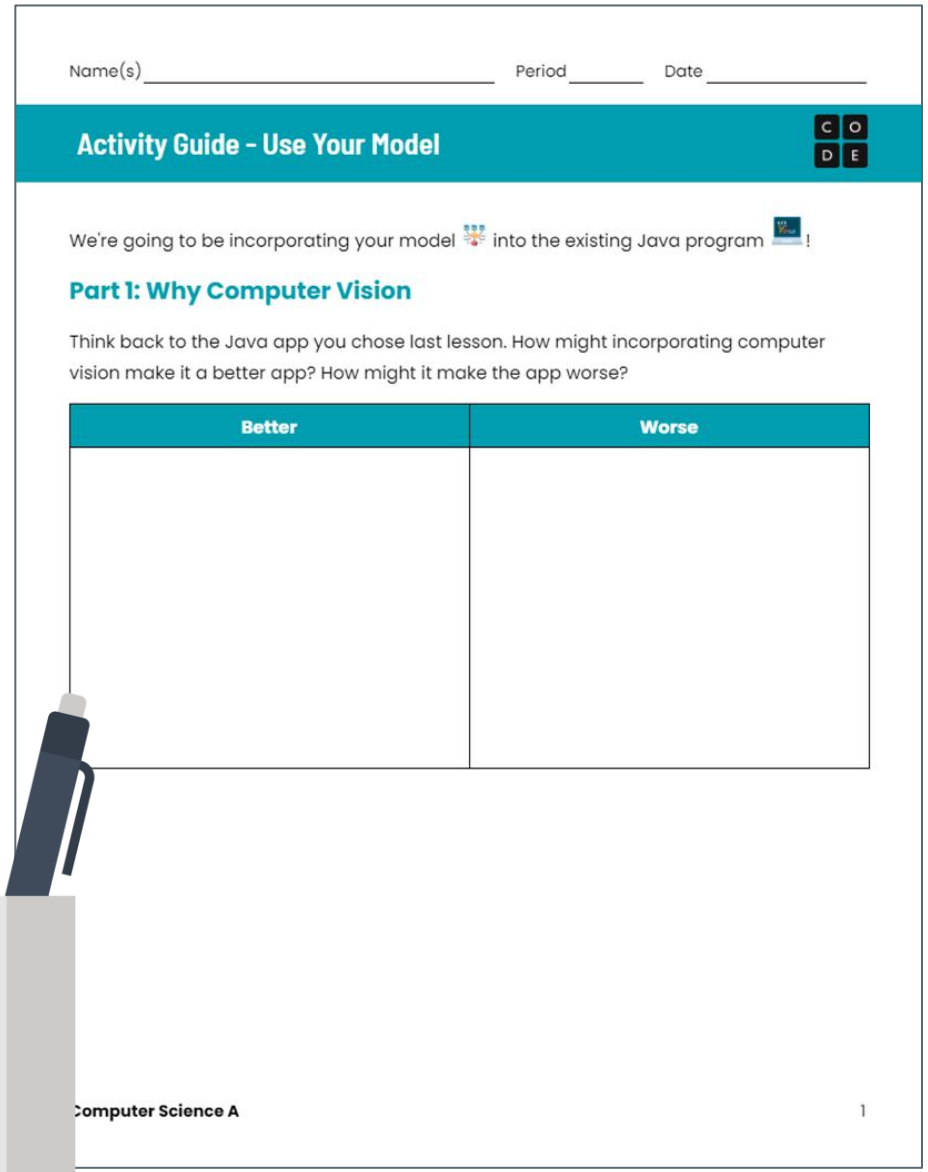
Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

Computer Science A 1

☑ Do This:

Try your partner's app and complete the **Your Partner's App** section of the activity guide.



Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

Better	Worse

Computer Science A 1

When To Use Computer Vision

"Sometimes you need to make sure there aren't **disparate error rates** across subgroups (e.g. melanoma detection).

Sometimes the task just **should not exist** (e.g. automatic gender recognition).

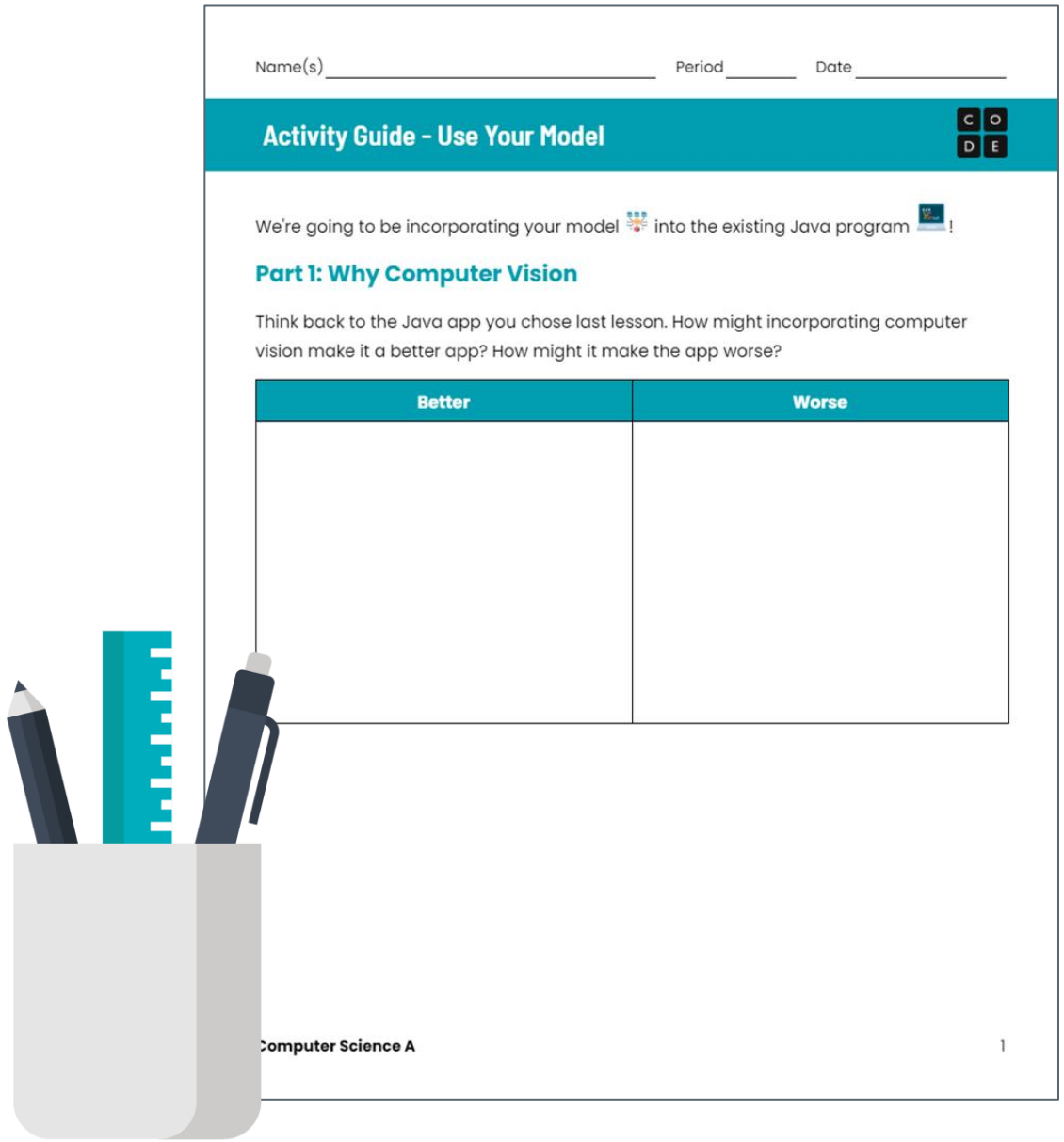
Sometimes the manner in which the tool is used is very problematic because of who has the **power (data)** and ability to train powerful models, vs who is subjected to those models."

- Remi Denton & Timnit Gebru, computer scientists

Tutorial on Fairness Accountability Transparency and Ethics in Computer Vision at CVPR 2020

☑ Do This:

Complete the **Software Engineering and Computer Vision** section of the activity guide.


Name(s) _____ Period _____ Date _____

Activity Guide - Use Your Model

We're going to be incorporating your model into the existing Java program !

Part 1: Why Computer Vision

Think back to the Java app you chose last lesson. How might incorporating computer vision make it a better app? How might it make the app worse?

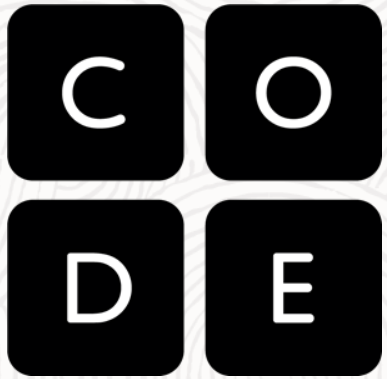
Better	Worse

Computer Science A 1

💡 Discuss:

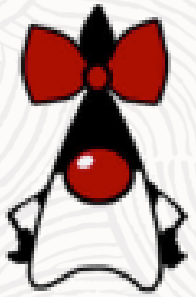
Given the previous quote and what you wrote down, when do you think software engineers **should or should not use** computer vision?

Foundational Java programs to learn, share, and collaborate



OpenJDK

ORACLE Academy



ORACLE UNIVERSITY

High quality information from the source

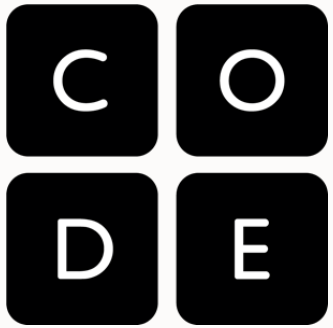
Dev.java

Inside.java

youtube.com/java

Java™

Thank you



https://dev.java

https://inside.java

https://youtube.com/java

https://github.com/java

https://code.org

https://engineering.tamu.edu/cse

 @Java | @OpenJDK

String Templates – Use in AP Computer Science A

In the 2022 Free Response Question 2, the `getBookInfo()` method in the `Textbook` class could be re-written from:

```
public String getBookInfo(){
   String bookinfo = super.getBookInfo();
   return bookinfo + "-" + edition;
}
```

to the following that uses `String` templates:

```
public String getBookInfo(){
   String bookinfo = super.getBookInfo();
   return STR. "\{bookinfo}-\{edition}";
}
```


Java™