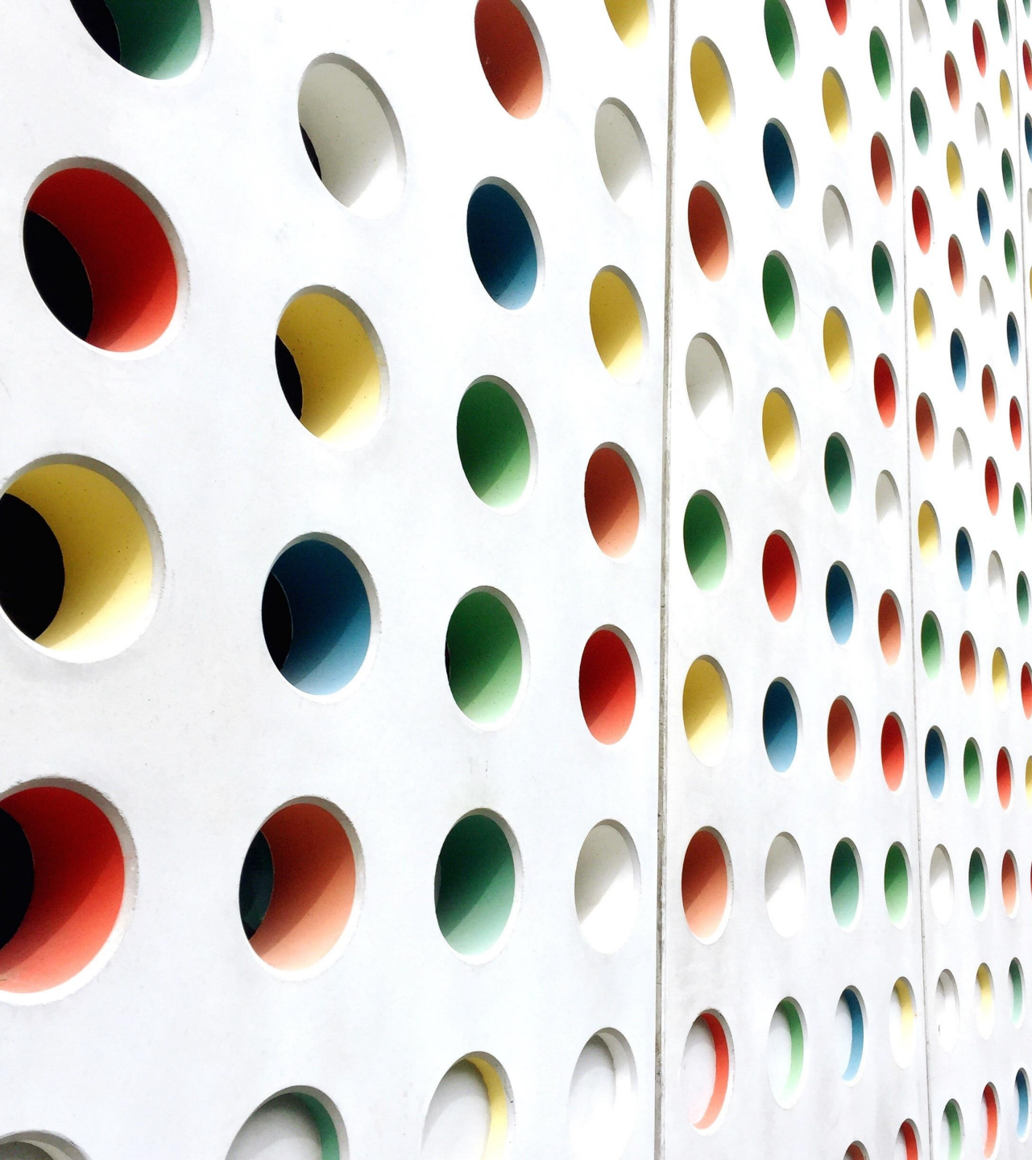


FIRST Tech Challenge — Java Beyond AP CSA

Introduction

- Courses that prepare students for the AP CSA exam cover a substantial amount of material about the Java programming language and its ecosystem.
- But it's also interesting to look at the document, put out by the College Board, that lists language features "Not tested in the AP CSA Exam, but potentially relevant/useful".
- Furthermore, there are some features of Java and its libraries which can be very useful in FTC, but which are not even mentioned in this document.

Introduction [cont.]

Today we will look at two examples:

1] How you can use the Java enum class to eliminate errors in the handling of AprilTag id numbers.

2] How you can use the java.currency package in Autonomous to move your robot across the field and simultaneously raise an elevator or arm.

Both examples include a lot of new concepts and vocabulary. They are meant to inspire curiosity.

All source code is freely available as shown in Appendix A.

If we have time we'll look at a third example

How can we get raw webcam frames out of the FTC VisionPortal and feed them into OpenCV?

This example introduces producer-consumer concurrency via the use of the Java classes ReentrantLock, Condition, and CountDownLatch.

Not tested in the AP CSA Exam, but potentially relevant/useful

We'll take a quick look at the document from the College Board that describes which Java constructs are included in the AP CSA exam and which are not.

https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-a-java-subset.pdf

Themes

Use the latest version of Java supported by Android Studio to aid in the development of software for FTC.

Practice good housekeeping – not just in naming conventions, factoring common code into methods, maximal use of the “private” keyword, and the organization of your code into packages, but in observing the “fail fast” principle, i.e. catching errors sooner rather than later.

Java has not stood still

This familiar example of Java's verbosity:

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

Can now be replaced by this:

```
void main() {
    System.out.println("Hello World!");
}
```

Where is Java?

CenterStage is at Android API level 24 and Java level 8.

Java 8 was released in 2014; the current level is Java SE 22.0.1.

Android 14 (API 34), the most recent, is at Java level 17.

We'll see which version of Android and Java will be supported for the 2024 – 2025 game.

For more information on what modern Java can do, look at the book ***Java Coding Problems – Second Edition*** (Packt).

Beyond Java AP CSA - Enumeration

The AprilTagProcessor in the FTC VisionPortal API includes a method getDetections() that returns a List<AprilTagDetection>. Each element of the List contains an integer id field of a detected AprilTag.

How can we keep track of these numeric ids?

Start with Item 34 in the book “Effective Java” by Joshua Bloch, formerly the Chief Java Architect at Google, “Use enums instead of int constants”.

We’ll walk through an AprilTag enum class that correlates numeric AprilTag ids with enum values and introduces the Java Collection EnumSet<E>, the Java Streams API, and Optional<T>, which is designed to cut down on null pointer exceptions.

Enum walkthrough [1]

Before enum types were added to JAVA, a common pattern for representing enumerated types was to declare a group of named int constants, one for each member of the type:

public static final int APPLE_FUJI = 0;

public static final int APPLE_PIPPIN = 1;

public static final int ORANGE_NAVEL = 0;

public static final int ORANGE_BLOOD = 1;

Enum walkthrough [2]

This technique, known as the “int enum pattern”, provides nothing in the way of type safety. The compiler won’t complain if you pass an apple to a method that expects an orange, compare apples to oranges with the == operator, or worse:

```
// Tasty citrus flavored applesauce!
int i = (APPLE_FUJI - ORANGE_NAVAL) / APPLE_PIPPIN;
```

Enum walkthrough [3]

Let's look at how we might use the built-in enum class for AprilTags:

```
public enum AprilTagId {
    TAG_ID_1, TAG_ID_3, TAG_ID_2 // order purposely reversed
}
```

The AprilTagProcessor in the FTC VisionPortal API includes a method getDetections() that returns a List<AprilTagDetection>. Each element of the List contains an integer id field of a detected AprilTag. How do we convert the numeric id to an enum value?

Enum walkthrough [4]

We have to modify the AprilTag enum class to associate a numeric value with each enum value:

```
public enum AprilTagId {
    TAG_ID_1(1), TAG_ID_3(3), TAG_ID_2(2)
}
```

So now what is full declaration of our AprilTag enum class? Let's look at the AprilTagUtils class in the IntelliJ sample project.

Enum walkthrough [s]

We need a method that will return an enum value given its numeric id.

```
public static AprilTagUtils.AprilTagId getEnumValue(int pNumericId) {
        AprilTagUtils.AprilTagId[] tagValues = AprilTagUtils.AprilTagId.values();
        for (AprilTagUtils.AprilTagId tagValue : tagValues) {
            if (tagValue.numericAprilTagId == pNumericId)
                return tagValue;
        }

        // No match.
        throw new AutonomousRobotException(TAG, "Invalid AprilTag number " + pNumericId);
    }
```

Enum walkthrough [6]

But here's a more modern way of finding the right enum value.

```
// Given the numeric id of an AprilTag return its enumeration.
public static AprilTagId getEnumValue(int pNumericId) {
   Optional<AprilTagId> matchingTag = EnumSet.allOf(AprilTagId.class).stream()
                       .filter(tag -> tag.getNumericId() == pNumericId)
                       .findFirst();

    return matchingTag.orElseThrow(() ->
       new AutonomousRobotException(TAG, "Invalid AprilTag number " +
                                    pNumericId));
}
```

Enum walkthrough [7]

A lot of what you see here is probably unfamiliar.

```
Optional<AprilTagId> matchingTag = // the Optional<T> class in
java.util
EnumSet.allOf(AprilTagId.class) // from the Collections Framework
.stream().filter(/* with lambda syntax */).findFirst() // from the
Streams API
```

Working backwards, the Streams API gives you a way of iterating through a collection of data without a “for” loop – and a lot more.

Enum walkthrough - Summary

Is this a lot of work to prevent unlikely errors?

In the case of AprilTags maybe yes, but you will have learned some useful techniques along the way and next time you'll be on the lookout for cases where "you need to represent a fixed set of constants**"**.

It is very worthwhile to look at the Streams API. There is a representative example from Chapter 4 of *Java in Action* in the IntelliJ sample project as StreamBasic.java. This example shows you how to traverse a collection without loops.

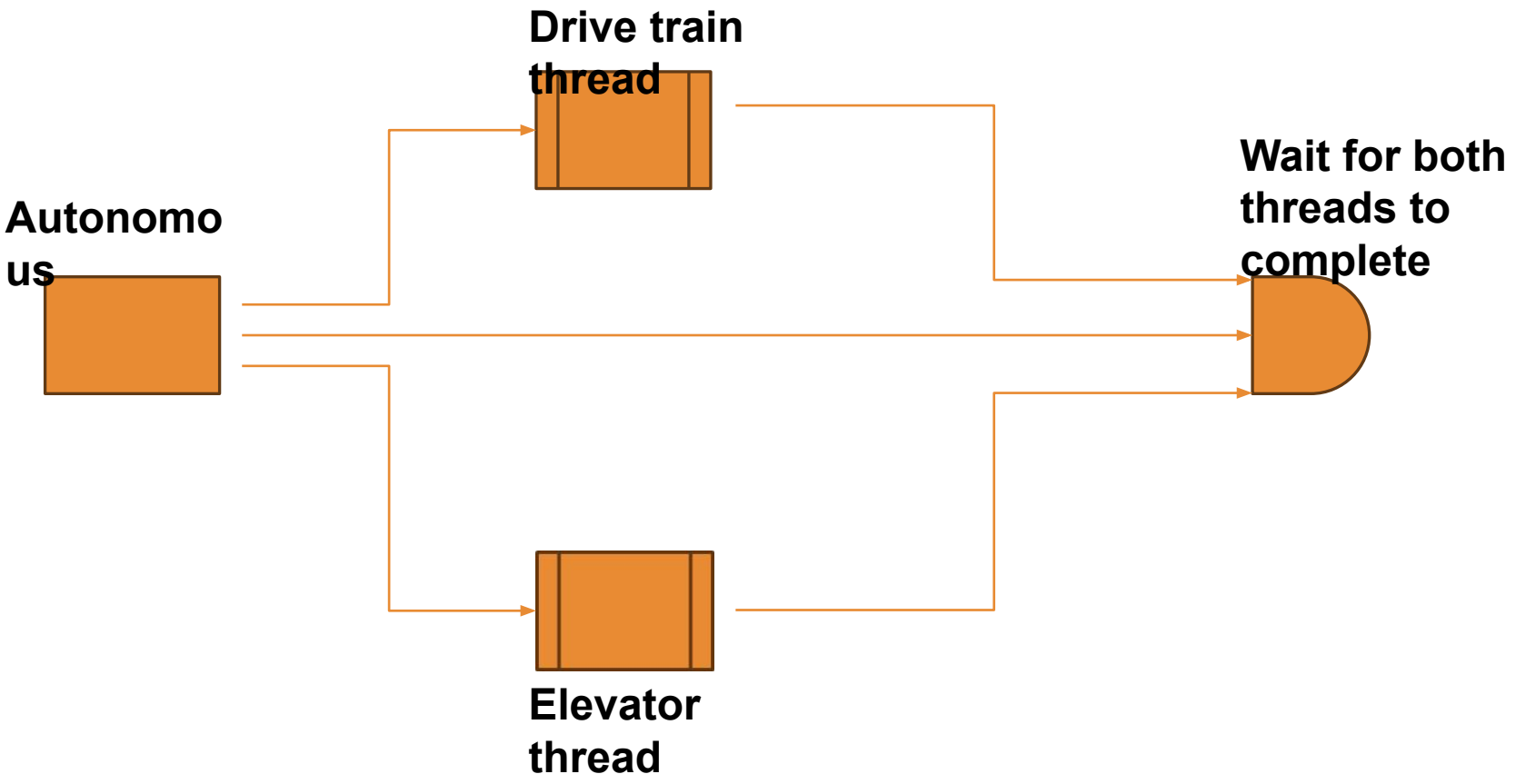
Java Concurrency

In Autonomous you want to move your robot and raise its elevator at the same time.

Since Java 8 the way to do this is to use Java Concurrency as provided in the ***java.util.concurrent*** package.

In particular you will combine the classes `Callable<V>` and `CompletableFuture<T>`.

Java Concurrency (2)



Java Concurrency [3]

Start with a method that moves the robot, for example:

```
straightLineMotion(targetClicks, angle, velocity);
```

Where “angle” is 0.0 (forward), -180.0 (back), 90.0 (strafe left), or -90.0 (strafe right).

Java Concurrency (4)

In cookbook fashion wrap the call to the method in a Java `Callable<V>`:

```
Callable<Void> callableDriveToPosition = () -> {
        straightLineMotion(1560, 90.0, .75);
        return null;
};
```

A **`Callable<V>`** is a Java functional interface (it has a single method). You can define an instance of a **`Callable<V>`** using lambda syntax. A **`Callable<V>`** is intended to be executed later, in this case in a separate thread.

Java Concurrency [5]

Also wrap the method moveElevator in a `Callable<V>`:

```
Callable<ElevatorLevel> callableMoveElevator = () -> {
return moveElevator(ElevatorLevel.LEVEL_2);
 };
```

The method moveElevator returns the level that the elevator actually achieved.

Java Concurrency [6]

```
// Launch a CompletableFuture thread for each callable:
CompletableFuture<Void> asyncMoveRobot =
Threading.launchAsync(callableDriveToPosition);

CompletableFuture<ElevatorLevel> asyncMoveElevator =
Threading.launchAsync(callableMoveElevator);


// Wait for both threads to complete:
Threading.getFutureCompletion(asyncMoveRobot);
ElevatorLevel actualLevel =
Threading.getFutureCompletion(asyncMoveElevator);
```

Java Concurrency (7)

These built-in threading classes are flexible in their application. You can add a third thread very easily, for example, if you need to move the robot, raise the elevator, and extend an arm all at the same time.

Look at FTCAutoThreading.java in both sample projects: IntelliJ and Android Studio.

Java Concurrency – Next Steps

For an overview of the ***java.util.concurrent package*** look at https://www.baeldung.com/java-concurrency.

Add test programs to the IntelliJ sample project and explore!

For example: Semaphore, CountDownLatch, CyclicBarrier.

Books

Effective Java (Addison-Wesley)

Head First Java (O'Reilly)

Modern Java in Action (Manning)

Java Coding Problems – *Second Edition* (Packt)

Websites

GeeksForGeeks

Baeldung

Jenkov

Mkyong

Stackoverflow

Callicoder (CompletableFuture)

Good Housekeeping

Not tested in the AP CSA Exam but useful in FTC:

- Keyword “final”
- visibility (public, private, package private, protected). Look at the FTC SDK examples ConceptAprilTag (which uses the “private” keyword) vs ConceptExploringIMUOrientation (which does not).
- **Pay attention to compiler warnings**, e.g. “Statement always evaluates to true”.
- Use Objects.requireNonNull, e.g. ArrayList<E> get(int index). See https://stackoverflow.com/questions/45632920/why-should-one-use-objects-requirenonnull.

A word on inheritance and recursion

These are important to learn and are covered in AP CSA.

However, look at ***Effective Java*** by Joshua Bloch, Item 16: Favor composition over inheritance.

Recursion is common in text parsing – in robotics not so much. If you want to practice recursion, take a look at a functional language such as Scala:

```
// Factorial function definition
 def fact(n:Int): Int=
 {
     if(n == 1) 1
     else n * fact(n - 1)
 }
```

Appendix A: Source Code

All source code is freely available in two sample projects:

1. Project IJFtcKickoff07Sep24 in IntelliJ – where you can test without the Robot Controller and Driver Station and use the debugger: https://github.com/NDHSRK/IJFtcKickoff07Sep24
2. Project FtcKickoff07Sep24 in Android Studio – where you can test in an FTC environment with the 9.2 release of the SDK: https://github.com/NDHSRK/FtcKickoff07Sep24

Appendix B: Get raw webcam frames from the VisionPortal

In the Android Studio sample project look at the TeleOp OpMode **WebcamFrameCapture** and the VisionPortal processor that it references, **RawFrameProcessor**.

The RawFrameProcessor also makes use of the Java concurrency classes CountDownLatch, ReentrantLock, and Condition.