# frgaal - a retrofit compiler for Java

The aim of frgaal is to make many of the latest features and enhancements to the Java language available on older runtimes. It enables you to compile code like this to run on a Java 8 JRE:

```
import java.util.List;
import java.util.Arrays;

public class Main {
    private static List<Integer> useVar() {
        var list = Arrays.asList(6, 1, 3, 5);
        useTextBlock(list);
        return list;
    }

    private static void useTextBlock(List<Integer> list) {
        String text = """
            initial list content
            is...""";
        System.err.println(text + list);
    }


    private static Result useInstanceOf(List<?> list) {
        final Object element = list.get(1);
        if (element instanceof Integer number) {
            return useSwitchExpr(number);
        }
        return new Result(element, "not a number!");
    }

    private static Result  useSwitchExpr(int number) {
        return switch (number) {
            case 3 -> new Result(number, "ok");
            default -> new Result(number, "bad");
        };
    }

    public static void main(String... args) {
        List<Integer> list = useVar();
        list.sort(null);
        System.err.println("after sorting: " + list);
        Result testResult = useInstanceOf(list);
        System.err.println(testResult.value() + " is " + testResult.comment());
    }

    public record Result(Object value, String comment) {}
}
```

Frgaal gives you the syntax of modern Java language, but unlike `javac` in the JDK, it allows you to compile them to run on Java 8.

## Supported Features

When targetting Java 8 or later (i.e. using `-target 8` or later), all Java 8 language features are supported. In addition following features are supported:

- `var` local variables, introduced in Java 10. Use `-source 10` or higher to enable.
- switch expressions, introduced in Java 14. Use `-source 14` or higher to enable.
- text blocks, permanent since Java 15. Use `-source 15` or higher to enable
- pattern matching in `instanceof`, permanent since Java 16. Use `-source 16` or higher to enable
- record classes, permanent since Java 16. Use `-source 16` or higher to enable
- sealed classes, permanent since Java 17. Use `-source 17` or higher to enable
- pattern matching for `switch`, introduced as preview in JDK 17, and updated in JDK 18, 19 and 20. Use `-source 20 --enable-preview` to enable
- `record` patterns, introduced as preview in JDK 19, and updated in JDK 20. Use `-source 20 --enable-preview` to enable

See "Preview Features" section below for more details.

## Usage with Maven

To use this compiler, specify following in your `pom.xml` file build section:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <dependencies>
                <dependency>
                    <groupId>org.frgaal</groupId>
                    <artifactId>compiler-maven-plugin</artifactId>
                    <version>20.0.0</version>
                </dependency>
            </dependencies>
            <configuration>
                <compilerId>frgaal</compilerId>
                <source>20</source>
                <target>1.8</target>
                <compilerArgs>
                    <arg>-Xlint:deprecation</arg>
                    <arg>--enable-preview</arg> <!--only needed when using preview languag
                </compilerArgs>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>3.2.0</version>
            <configuration>
                <archive>
                    <manifestEntries>
                        <Multi-Release>true</Multi-Release>
                    </manifestEntries>
                </archive>
            </configuration>
        </plugin>
    </plugins>
</build>
```

With such a change the compiler of your project no longer depends on the used JDK. All the compiler code is downloaded from Maven central and it can run on anything from JDK8 up. If you want to update your compiler to get a bugfix or to use latest language feature, you can change to some newer version. However, until you do that, no matter what breaking changes appear in the JDK, your project is still going to compile into exactly the same `.class` files.

## Usage with Gradle DSL style

You need gradle 6.8.x to be able to use this compiler. To use this compiler, specify following in your `build.gradle` file:

```
plugins {
    id 'java'
    id 'org.frgaal.compiler'
}

targetCompatibility = '1.8'
sourceCompatibility = '20'

compileJava {
    options.compilerArgs << '-Xlint:deprecation' << '--enable-preview'
}

compileTestJava {
    options.compilerArgs << '-Xlint:deprecation' << '--enable-preview'
}
```

To be able to resolve the frgaal compiler plugin you need to specifiy the following in your `settings.gradle` file, at the top:

```
pluginManagement {
    resolutionStrategy {
        eachPlugin {
            if (requested.id.namespace == 'org.frgaal') {
                useModule('org.frgaal:compiler-gradle-plugin:<Version>')
            }
        }
    }
    repositories {
        maven {
            url 'https://mvnrepository.com/artifact/org.frgaal'
        }
        gradlePluginPortal()
    }
}
```

## Usage on command line

To use the frgaal compiler, run it as follows:

```
$ java -jar compiler.jar <javac-parameters>
```

The main entrypoint is `org.frgaal.Main` class.

## Preview Features

Frgaal compiler supports *preview features* of the Java language. Some of the *preview features* can be used with `--target 1.8`. Namely:

- pattern matching for `switch`
- `record` patterns

These features require additional standard command line parameters

```
$ java -jar compiler.jar --enable-preview -source <latest_version> <javac-parameters>
```

The classfiles produced by frgaal compiler are usable on the target Java version, or newer Java versions. Source code using preview features is subject to future Java language standardization and may need adjustments when switching to newer versions of the frgaal compiler.

## System Paths

By default, the frgaal compiler uses system classes that correspond to the specified target platform version. Regardless of the platform on which the frgaal compiler runs, APIs from the target platform are available. Even if you run the compiler on JDK8, you can use JDK11 APIs in your `Code.java` when specifying `-target 11`:

```
$ cat >Code.java <<END
class Code {
  public static void main(String[] args) {
    var unavailableOnJDK8 = java.lang.Module.class;
    System.out.println(unavailableOnJDK8.getName());
  }
}
END
$ jdk1.8.0/bin/java -jar compiler.jar -source 20 -target 11 Code.java
```

To disable this behavior use `-bootclasspath`, `-Xbootclasspath` or `-system` and specify the desired target platform API. Or use `-XDignore.symbol.file` to disable this behavior and use runtime platform APIs.

## Caveats

The current caveats include:

- module-info.java cannot be compiled with `--target 8`
- sealed classes are only enforced at runtime when running on Java 17 or newer
- the pattern matching feature does not use the `java.lang.MatchException`, introduced as a preview API in JDK 19. `java.lang.IllegalStateException` is used to wrap exceptions from record accessors, and `java.lang.IncompatibleClassChangeError` is used when the hierarchy changes so that there is not match for an exhaustive switch.

### Record Classes

The record classes depend on a number of runtime features, and are impossible to fully support for older versions of the platform. frgaal will allow to compile record classes with the following caveats:

- `Serializable` record classes are only supported with `--target 16` and later.
- for targets below 16, the super class of the record class is `java.lang.Object`, and the support for record classes in the standard libraries is missing. Some serialization libraries may not recognize the class as a record class.
- the frgaal compiler will produce a separate class file of the record class under `META-INF/versions/16`, which will have the proper superclass and other runtime support. If the classfiles are packed into a jar marked as a multi-release jar, the runtime may be able to use the JDK 16 version with the better runtime support.

### Multi-Release JARs

Several features (like the record classes, or sealed classes) require runtime support, which is not available on older platforms. frgaal will generate multiple versions of the classfiles, for use in older versions of the Java platform, which may lack some features, and fully featured for use in newer versions of the Java platform. The structure matches the multi-release jar structure. When the classes are packed in a jar marked as a multi-release jar, the Java runtime will be able to use the appropriate version of the classfiles.

## Building

Run the `build.sh` script. The compiler will be in the `dist/compiler-*.jar` file.

## License

The license is GPLv2+CPE.