CraftCode

Our Craftsmanship defines your Code

Our Craftsmanship defines your code

SOFWARE CRAFTSMANSHIP – Craftsmanship manifesto

COMMUNITY – In our company and beyond

JAVA, SPRING, ANGULAR, REACT…– The tech stack

FUN – Karting, Dinner, Drinks and Ski trip

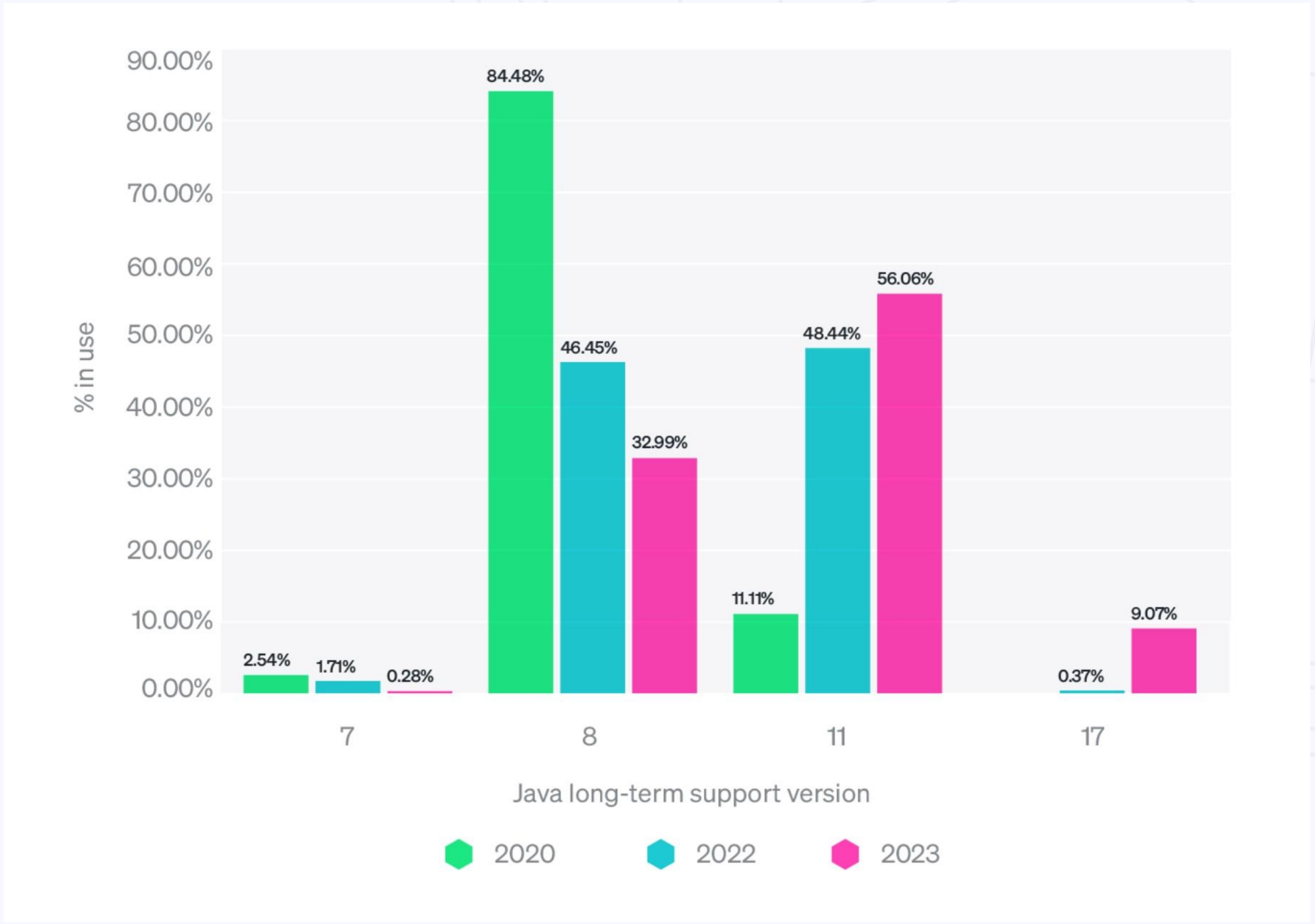
CONSULTANCY– Experts who add value

Learn to write code like a Craftsman

CraftCode nodigt je uit

CraftCode ACADEMY

Java Adoption



Source: New Relic

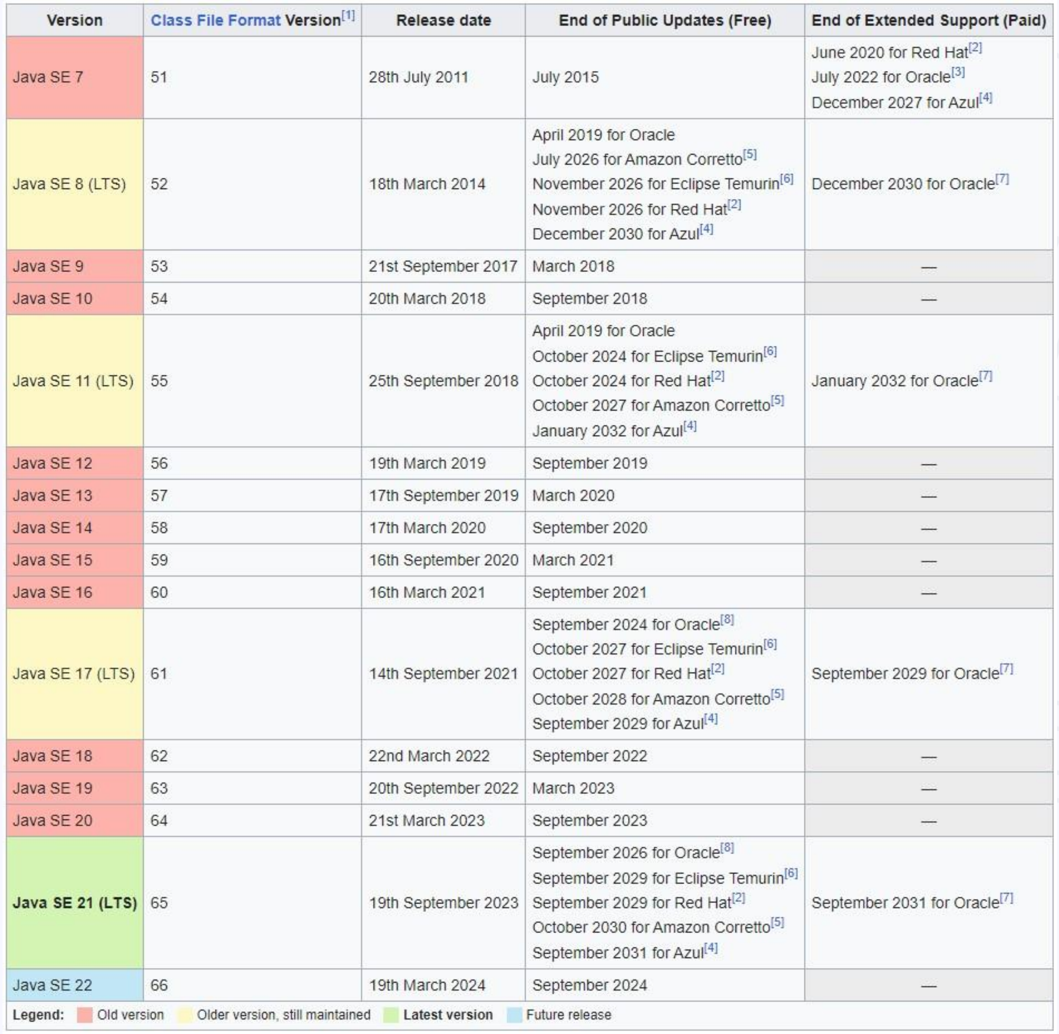
Why keep up to date?

Benefits
1. Use latest, greatest and safest libraries and frameworks
2. Security fixes and performance improvements
3. Better developer experience

Drawbacks
- Takes time away from creating value for the business and users
- Frameworks/tools do not support the new version

CraftCode Academy

Java Support



Version	Class File Format Version[1]	Release date	End of Public Updates (Free)	End of Extended Support (Paid)
Java SE 7	51	28th July 2011	July 2015	June 2020 for Red Hat[2] July 2022 for Oracle[3] December 2027 for Azul[4]
Java SE 8 (LTS)	52	18th March 2014	April 2019 for Oracle July 2026 for Amazon Corretto[5] November 2026 for Eclipse Temurin[6] November 2026 for Red Hat[2] December 2030 for Azul[4]	December 2030 for Oracle[7]
Java SE 9	53	21st September 2017	March 2018	—
Java SE 10	54	20th March 2018	September 2018	—
Java SE 11 (LTS)	55	25th September 2018	April 2019 for Oracle October 2024 for Eclipse Temurin[6] October 2024 for Red Hat[2] October 2027 for Amazon Corretto[5] January 2032 for Azul[4]	January 2032 for Oracle[7]
Java SE 12	56	19th March 2019	September 2019	—
Java SE 13	57	17th September 2019	March 2020	—
Java SE 14	58	17th March 2020	September 2020	—
Java SE 15	59	16th September 2020	March 2021	—
Java SE 16	60	16th March 2021	September 2021	—
Java SE 17 (LTS)	61	14th September 2021	September 2024 for Oracle[8] October 2027 for Eclipse Temurin[6] October 2027 for Red Hat[2] October 2028 for Amazon Corretto[5] September 2029 for Azul[4]	September 2029 for Oracle[7]
Java SE 18	62	22nd March 2022	September 2022	—
Java SE 19	63	20th September 2022	March 2023	—
Java SE 20	64	21st March 2023	September 2023	—
Java SE 21 (LTS)	65	19th September 2023	September 2026 for Oracle[8] September 2029 for Eclipse Temurin[6] September 2029 for Red Hat[2] October 2030 for Amazon Corretto[5] September 2031 for Azul[4]	September 2031 for Oracle[7]
Java SE 22	66	19th March 2024	September 2024	—

Legend: 🟥 Old version 🟨 Older version, still maintained 🟩 **Latest version** 🟦 Future release

Learn to write code like a Craftsman

What is a JEP?

JEP stands for: **J**DK (**J**ava **D**evelopment **K**it) **E**nhancement **P**roposal

jwebserver

A simple server included with java.

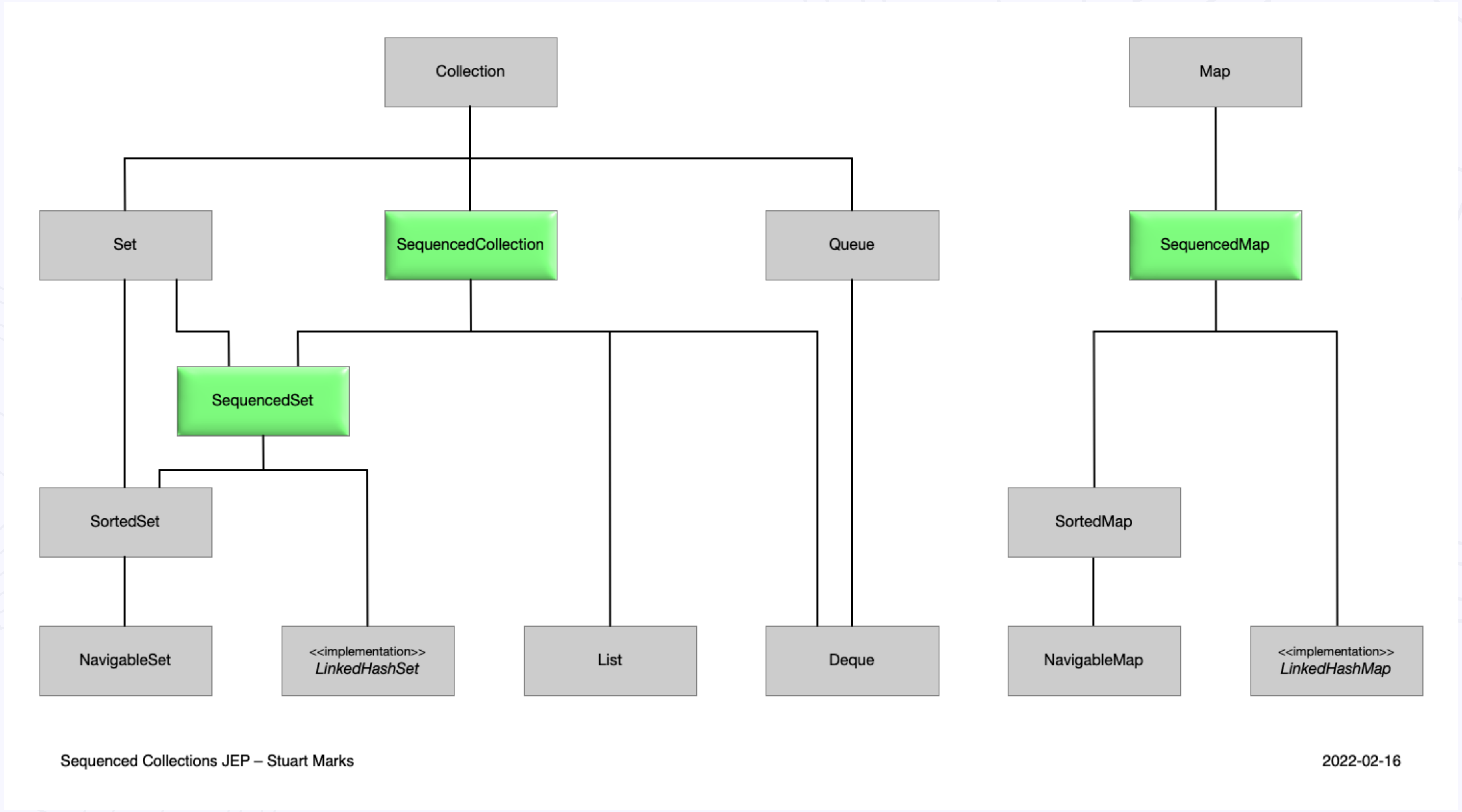
```
john@linux-desktop:~$ jwebserver

Binding to loopback by default. For all interfaces use "-b 0.0.0.0" or "-b ::".
Serving /jwebserver and subdirectories on 127.0.0.1 port 8000
URL http://127.0.0.1:8000/
127.0.0.1 - - [13/Nov/2023:16:29:50 +0100] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2023:16:29:50 +0100] "GET /favicon.ico HTTP/1.1" 404 -
```

Sequenced Collections

```
list.get(list.size() - 1)
deque.getLast()
sortedSet.last()
```

```
interface SequencedCollection<E> extends Collection<E> {
  SequencedCollection<E> reversed();
  void addFirst(E);
  void addLast(E);
  E getFirst();
  E getLast();
  E removeFirst();
  E removeLast();
}
```

Sequenced Collections



Pattern matching switch

```
// BEFORE JAVA 16
if (obj instanceof String) {
            String str = (String) obj;
            System.out.println(str);
}

// JAVA 16+
 if (obj instanceof String str) {
            System.out.println(str);
}
```

Pattern matching switch

```java
// BEFORE JAVA 21
static String asStringValue(Object anyValue) {
        String result = null;
        if (anyValue instanceof String str) {
                result = str;
        } else if (anyValue instanceof BigDecimal bd) {
                result = bd.toEngineeringString();
        } else if (anyValue instanceof Integer i) {
                result = Integer.toString(i);
        } else {
                result = "n/a";
        }
        return result;
}
```

```java
// JAVA 21+
static String asStringValue(Object anyValue) {
        return switch (anyValue) {
                case String str -> str;
                case BigDecimal bd -> bd.toEngineeringString();
                case Integer i -> Integer.toString(i);
                default -> "n/a";
        };
}
```

Record patterns

```
record Point(int x, int y) {}
```

```
//To match record, then access component:

Object maybePoint = ...;
if (maybePoint instanceof Point p) {
        System.out.println("Point => " + p.x() + "/" + p.y());
}
```

```
//With Java 21, use Record Pattern to access component
directly:

Object maybePoint = ...;
if (maybePoint instanceof Point(int x, int y)) {
        System.out.println("Point => " + x + "/" + y);
}
```

Record patterns

Benefits mainly in **nested** records:

```
record Size(int width, int height) { }
record Point(int x, int y) { }
record WindowFrame(Point origin, Size size) { }

//To match record, then access height component as before:
if (obj instanceof WindowFrame wf) {
        if (wf.size() != null) {
                System.out.println("Height: " + wf.size().height());
        }
}

//Much simpler when using record patterns:
if (obj instanceof WindowFrame(Point origin, Size(int width, int height))) {
        System.out.println("Height: " + height);
}
```

Data Oriented Programming

- Project Amber
- Separating code (behavior) from data.
- Representing data with generic data structures.
- Treating data as immutable.
- Separating data schema from data representation.

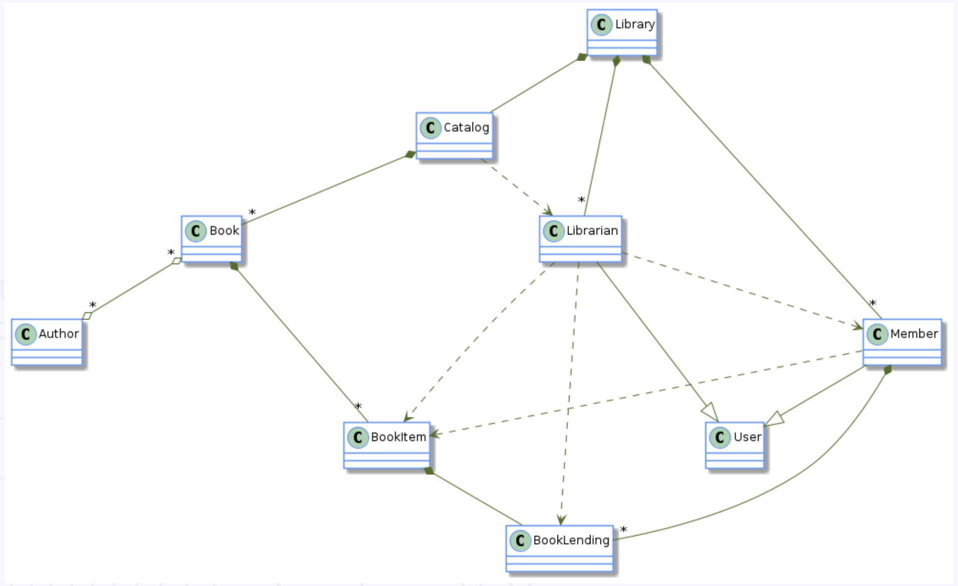
String Templates (Preview)

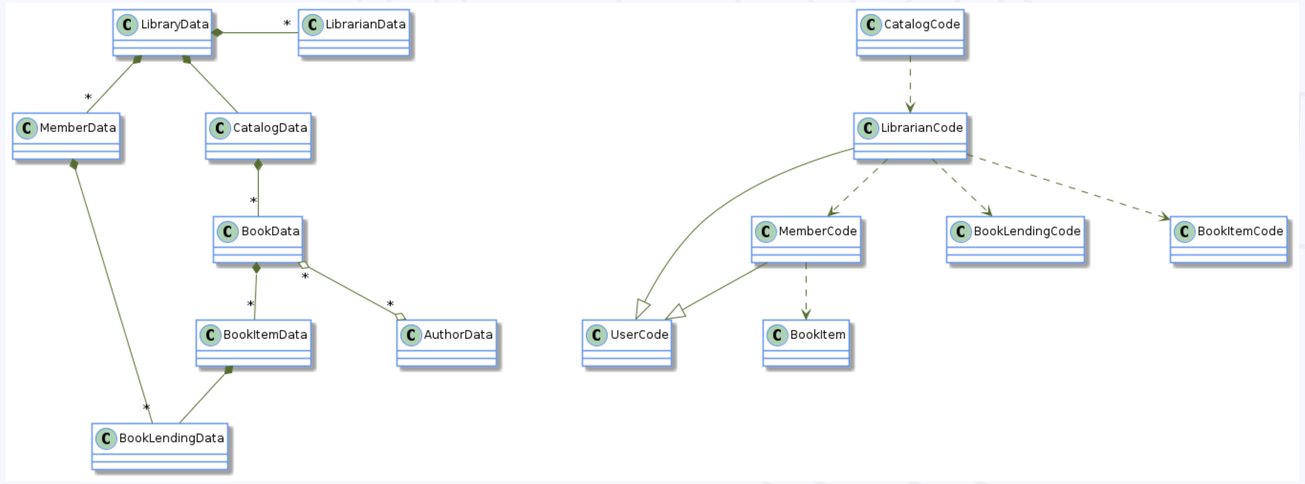
```
String greeting = "Howdy";
String person = "neighbor";
String location = "neighborhood";

// The + Operator
String plusConcat = greeting + " " + person + "! Welcome
to the " + location + "!";

// StringBuilder
String stringBuilderConcat = new StringBuilder()
    .append(greeting)
    .append(" ")
    .append(person)
    .append("! Welcome to the ")
    .append(location)
    .append("!")
    .build();
```

```
// String.format()
String stringFormatConcat =
String.format("%s %s! Welcome to the %s!", greeting,
person, location);
```

```
// NEW! String Interpolation with Template Expressions
String stringInterpolationConcat =

STR."\{greeting} \{person}! Welcome to the \{location}!";
```

Data Oriented Programming





Virtual Threads

Improvements
- 2000 – 5000 => 1.000.000+
- 6x faster

Consequences
- No more thread pools
- Different approach to async programming

Lab

https://gitpod.io/github.com/CraftCodeBE/java21-gitpod

https://github.com/CraftCodeBE/java21-gitpod.git

1. File > New > Project from Version Control…
2. In the URL field paste: https://github.com/CraftCodeBE/java21-gitpod.git and click on “Clone”:
3. Once the project is cloned and opened, checkout the “preview_enabled” branch:
4. Then if you don’t already have Java 21, go to File > Project Structure…
5. In Platform Settings > SDKs, click the plus sign and “Download JDK…”:
6. Download version 21 from vendor Eclipse Temurin:
7. Once it’s downloaded, In Project Settings > Project, change the SDK to temurin-21 and Language level to 21 (Preview):

CraftCode

Our Craftsmanship defines your Code