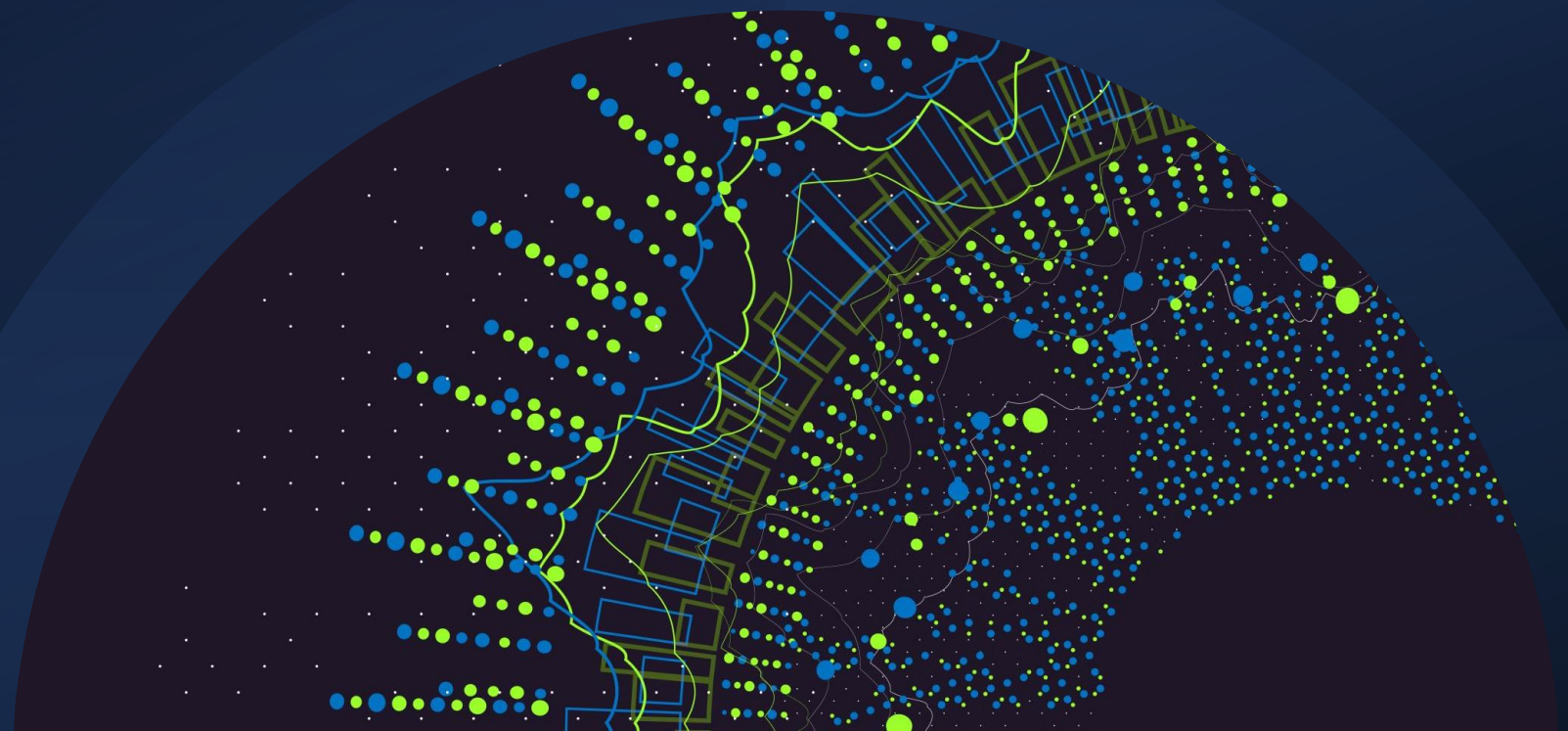
Enum



Enum – Enumerations

Spezielle Klasse, stellt eine Gruppe von Konstanten dar

Kann innerhalb einer Klasse deklariert und initialisiert werden

Kann wie eine normale Klasse Methoden und Attribute haben aber:

- Alles ist public static und final
- Kann Interfaces implementieren
- Kann nicht geerbt werden oder benutzt werden um Objekte zu erstellen

Jedes Aufzählungselement ist ein Objekt

Enum – Enumerations

Example

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
}

public class Main {
  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;

    switch(myVar) {
      case LOW:
        System.out.println("Low level");
        break;
      case MEDIUM:
         System.out.println("Medium level");
        break;
      case HIGH:
        System.out.println("High level");
        break;
    }
  }
}
```

The output will be:

```
Medium level
```

Enum – Wann und wieso?

- Eine Möglichkeit ungültige Variablen als Methodenparameter zu verhindern

Enum – Wann und wieso?

```
/** Counts number of foobangs.
 * @param type Type of foobangs to count. Can be 1=green foobangs,
 * 2=wrinkled foobangs, 3=sweet foobangs, 0=all types.
 * @return number of foobangs of type
 */
public int countFoobangs(int type)
```

versus

```
/** Types of foobangs. */
public enum FB_TYPE {
 GREEN, WRINKLED, SWEET,
 /** special type for all types combined */
 ALL;
}

/** Counts number of foobangs.
 * @param type Type of foobangs to count
 * @return number of foobangs of type
 */
public int countFoobangs(FB_TYPE type)
```

Enum – Wann und wieso?

Ein Methodenaufruf wie z.B.

```
int sweetFoobangCount = countFoobangs(3);
```

Wird zu:

```
int sweetFoobangCount = countFoobangs(FB_TYPE.SWEET);
```

- Sofort klar was als Parameter erlaubt ist
- Falscher Parameter wird vom Compiler erkannt

Bessere Lesbarkeit!