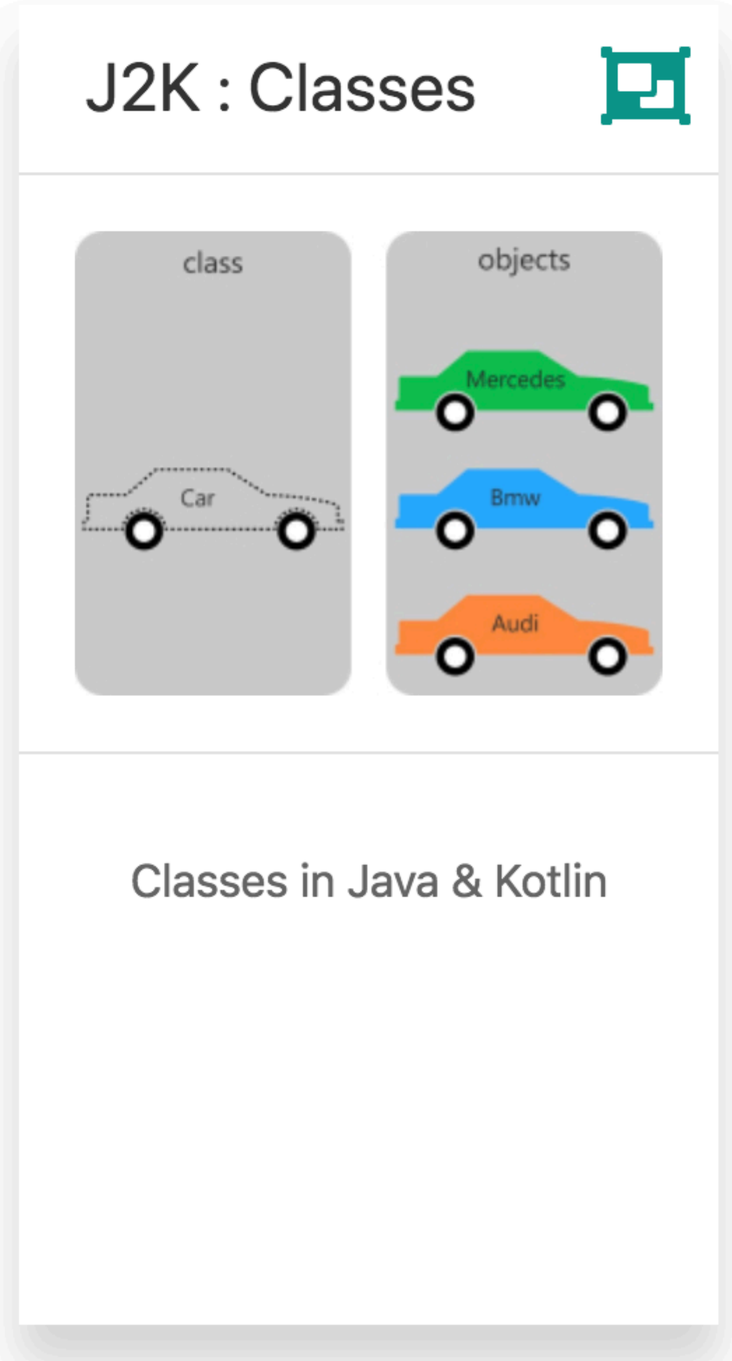
J2K: Classes


J2K : Classes
class
Car
objects
Mercedes
Bmw
Audi
Classes in Java & Kotlin

Java

```java
public class Utils {

    private Utils() {
      // This utility class is not publicly instantiable
    }

    public static int getScore(int value) {
        return 2 * value;
    }

}
```

Kotlin

```kotlin
class Utils private constructor() {

    companion object {

        fun getScore(value: Int): Int {
            return 2 * value
        }

    }
}

// another way

object Utils {

    fun getScore(value: Int): Int {
        return 2 * value
    }

}
```

Java

```java
public class Utils {

    private Utils() {
      // This utility class is not publicly instantiable
    }

    public static int getScore(int value) {
        return 2 * value;
    }

}
```

Kotlin

```kotlin
class Utils private constructor() {

    companion object {

        fun getScore(value: Int): Int {
            return 2 * value
        }

    }
}

// another way

object Utils {

    fun getScore(value: Int): Int {
        return 2 * value
    }

}
```

Java

```java
public class Developer {

    private String name;
    private int age;

    public Developer(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Developer developer = (Developer) o;

        if (age != developer.age) return false;
        return name != null ? name.equals(developer.name) : developer.name == null;

    }

    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
        result = 31 * result + age;
        return result;
    }

    @Override
    public String toString() {
        return "Developer{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

Kotlin

```
data class Developer(var name: String, var age: Int)
```

Java

```java
public class Developer implements Cloneable {

    private String name;
    private int age;

    public Developer(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        return (Developer)super.clone();
    }
}

// cloning or copying
Developer dev = new Developer("Mindorks", 30);
try {
    Developer dev2 = (Developer) dev.clone();
} catch (CloneNotSupportedException e) {
    // handle exception
}
```

Kotlin

```kotlin
data class Developer(var name: String, var age: Int)

// cloning or copying
val dev = Developer("Mindorks", 30)
val dev2 = dev.copy()
// in case you only want to copy selected properties
val dev2 = dev.copy(age = 25)
```

Java

```
Person person;
```

Kotlin

```
internal lateinit var person: Person
```

Java

```
Person person;
```

Kotlin

```
internal lateinit var person: Person
```

Java

```
if (object instanceof Car) {
}
Car car = (Car) object;
```

Kotlin

```kotlin
if (object is Car) {
}
var car = object as Car

// if object is null
var car = object as? Car // var car = object as Car?
```

Java

```java
if (object instanceof Car) {
}
Car car = (Car) object;
```

Kotlin

```kotlin
if (object is Car) {
}
var car = object as Car

// if object is null
var car = object as? Car // var car = object as Car?
```