INTRODUCTION TO JAVA
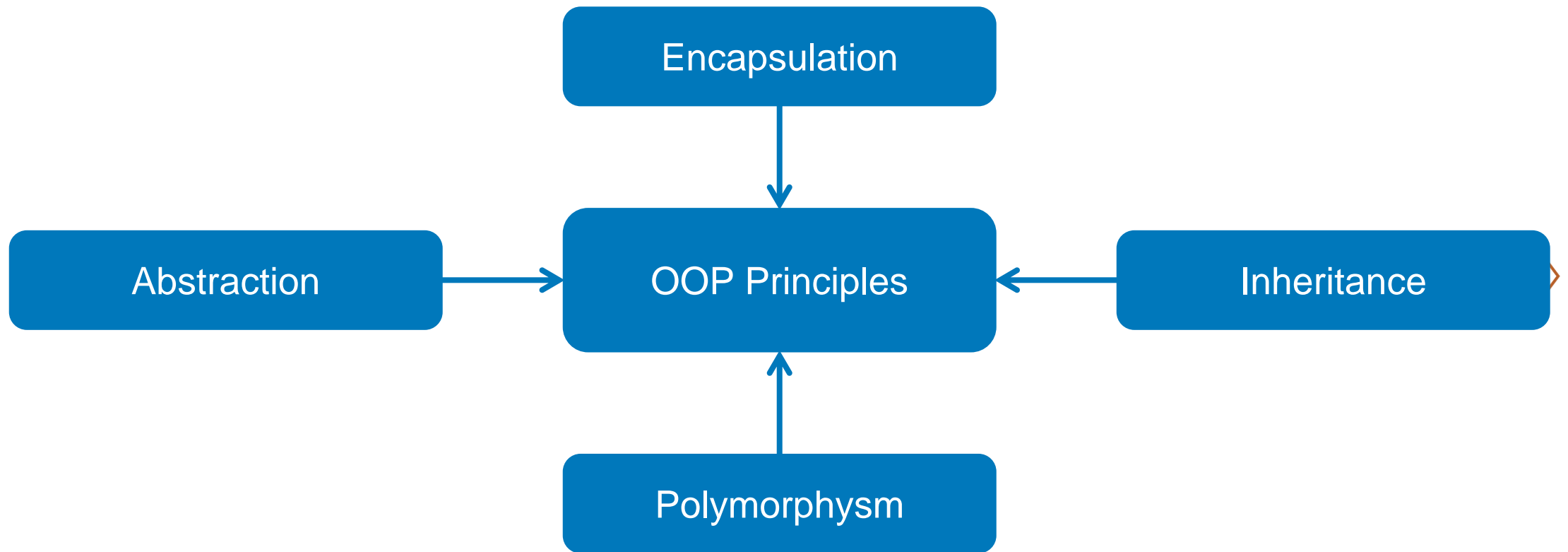
Java 1.0

ABSTRACTION

Lesson # 09

OBJECT-ORIENTED PROGRAMMING PRINCIPLES
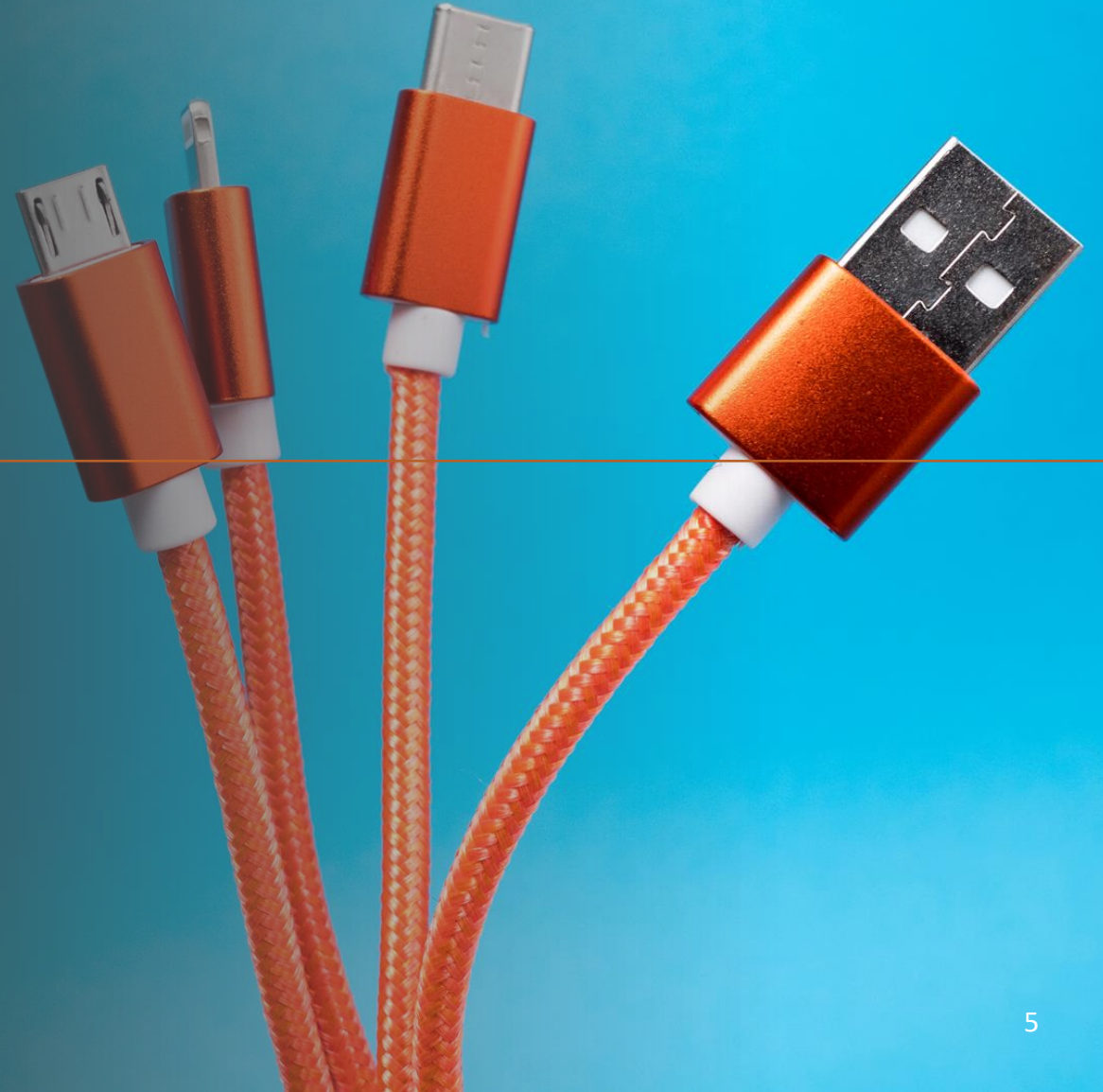
PILLARS OF OBJECT-ORIENTED PROGRAMMING

ABSTRACTION

ABSTRACTION OVERVIEW

- The process where you show only relevant data and hide unnecessary details of an object from user

- Allows you to abstract from usage and rather outline generic object functionality

- Defines what object does instead of how

JAVA ABSTRACTION

- Abstraction is achieved by two mechanisms:

  - Interfaces

    - Allows to achieve complete abstraction

  - Abstract classes

    - Allows to achieve partial abstraction

JAVA INTERFACES

- A bit like class, except:

  - Interface can only contain method signatures and fields

- Methods defined in interfaces cannot contain the implementation of method, only signature (return type, name, parameters, exceptions)

- Describes an object by actions it can perform

- Sometimes interface names end with '-able' postfix (e.g. comparable)

JAVA INTERFACE EXAMPLE

Interface name

Interface keyword instead of class

```java
public interface Singer {

    void sing();

}
```

Singers can sing, but we don't care how they do so

JAVA INTERFACE EXAMPLE

```java
public class ElvisPresley implements Singer{

    @Override
    public void sing() {
        System.out.println("Love me tender, baby...");
    }

}
```

Special keyword to guarantee that we support interface specified behavior

Concrete implementation of singers behavior

JAVA INTERFACE EXAMPLE

```java
public class MichaelJackson implements Singer {

    @Override
    public void sing() {
        System.out.println("Billie Jean is not my lover...");
    }

}
```

JAVA INTERFACE EXAMPLE

```java
public class BritneySpears implements Singer {

    @Override
    public void sing() {
        System.out.println("Hit me baby one more time...");
    }

}
```
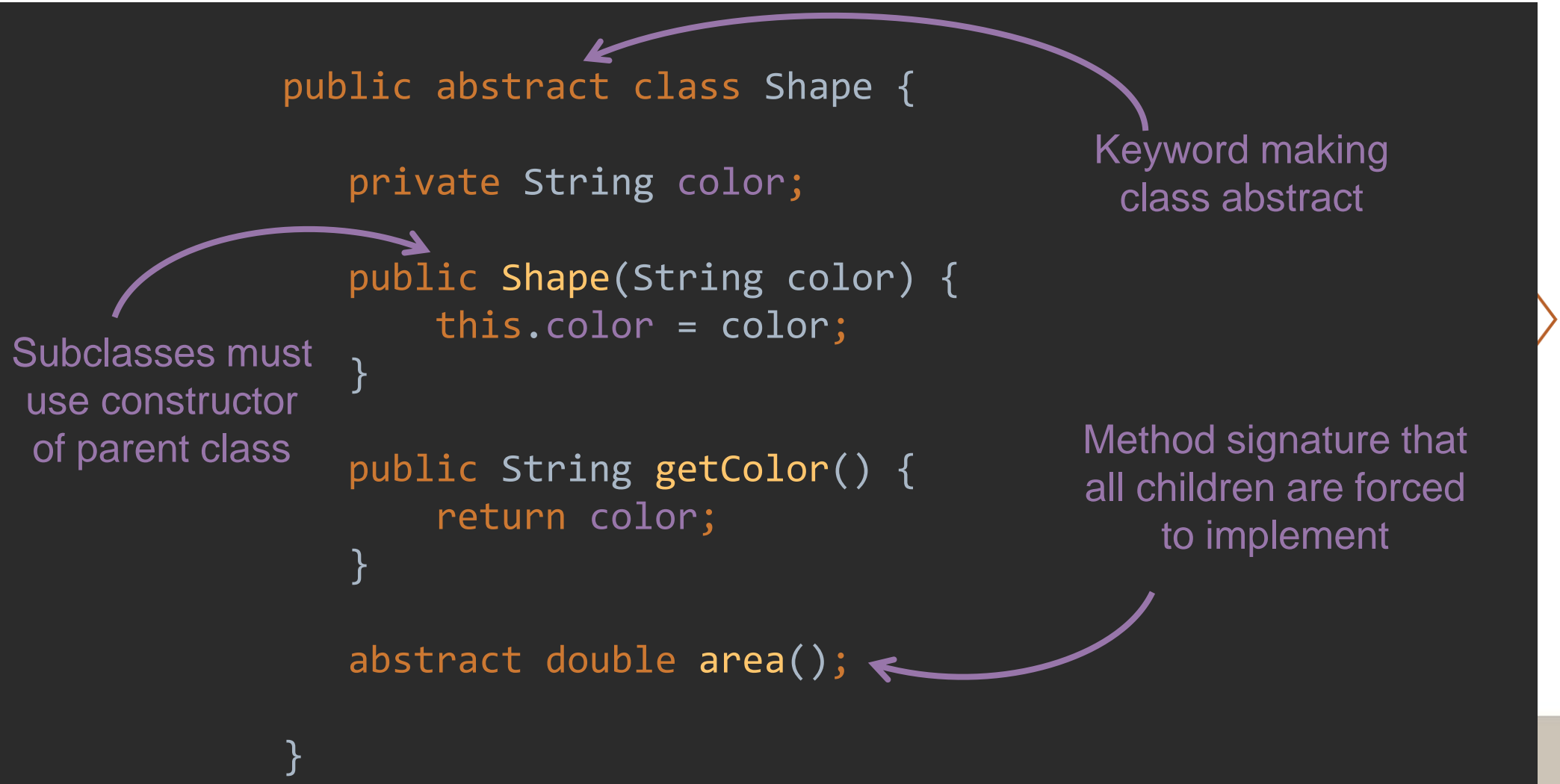
JAVA ABSTRACT CLASSES

- Mostly like a class, except:

  - Can contain method signatures without implementation among other methods

  - Cannot be instantiated

JAVA ABSTRACT CLASS EXAMPLE

```java
public abstract class Shape {

    private String color;

    public Shape(String color) {
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    abstract double area();

}
```

Keyword making class abstract

Subclasses must use constructor of parent class

Method signature that all children are forced to implement

JAVA ABSTRACT CLASS EXAMPLE

```java
public class Circle extends Shape {

    private int radius;

    public Circle(String color, int radius) {
        super(color);
        this.radius = radius;
    }


    @Override
    double area() {
        return 3.14 * radius * radius;
    }
}
```
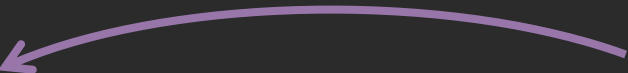
Circle specific properties

Extending shape class with concrete details

Calling parent constructor with required params

Each concrete shape knows how calculate its area

JAVA ABSTRACT CLASS EXAMPLE

```java
public class Rectangle extends Shape {

    private int width;
    private int height;


    public Rectangle(String color, int width, int height) {
        super(color);
        this.width = width;
        this.height = height;
    }


    @Override
    double area() {
        return width * height;
    }
}
```

Rectangles specific properties

INTERFACE VS ABSTRACT CLASS

- Type of methods

  - Interface can have only abstract methods (since Java 8 supports static and default methods as well)

  - Abstract class can have abstract and non-abstract methods

- Final variables

  - Variables declared in a Java interface are by default final

  - Abstract class may contain non-final variables

INTERFACE VS ABSTRACT CLASS

- Type of variables

  - Interface has only static and final variables

  - Abstract class can have final, non-final, static and non-static variables

- Implementation

  - Interface can't provide the implementation of abstract class

  - Abstract class can provide the implementation of interface

INTERFACE VS ABSTRACT CLASS

- Inheritance vs Abstraction

    - Interface can be implemented using keyword "implements"

    - Abstract class can be extended using keyword "extends"

- Multiple Implementation

    - Interface can extend another Java interface only

    - Abstract class can extend another Java class and implement multiple Java interfaces

INTERFACE VS ABSTRACT CLASS

- Accessibility of data members

  - Access modifiers of interface members are public by default and cannot be changed

  - Access modifiers of abstract class members can have any access modifiers (except private abstract methods)

POLYMORPHISM

POLYMORPHISM

- Polymorphism is the ability of an object to take on many forms

- Capability of a method to do different things based on the object that it is acting upon

- Which implementation to be used is decided at runtime depending upon the situation

POLYMORPHISM EXAMPLE

Code

```
Singer elvis = new ElvisPresley();
Singer jackson = new MichaelJackson();
Singer spears = new BritneySpears();

elvis.sing();
jackson.sing();
spears.sing();
```

Console output

```
Love me tender, baby...
Billie Jean is not my lover
Hit me baby one more time
```

POLYMORPHISM EXAMPLE

Code

```java
Singer[] singers = new Singer[2];
singers[0] = new ElvisPresley();
singers[1] = new BritneySpears();

for (Singer singer : singers) {
    singer.sing();
}
```

Console output

```
Love me tender, baby...
Hit me baby one more time
```

POLYMORPHISM EXAMPLE

Code

```java
Shape circle = new Circle("Red", 3);
Shape rectangle = new Rectangle("Blue", 2, 4);

System.out.println("Circle area = " + circle.area());
System.out.println("Rectangle area = " + rectangle.area());
```

Console output

```
Circle area = 28.259999999999998
Rectangle area = 8.0
```

REFERENCES

REFERENCES

- https://stackify.com/oops-concepts-in-java/

- https://beginnersbook.com/2013/03/oops-in-javaencapsulation-inheritance-polymorphism-abstraction/

- https://www.geeksforgeeks.org/abstraction-in-java-2/

- http://tutorials.jenkov.com/java/interfaces.html

THANK YOU!