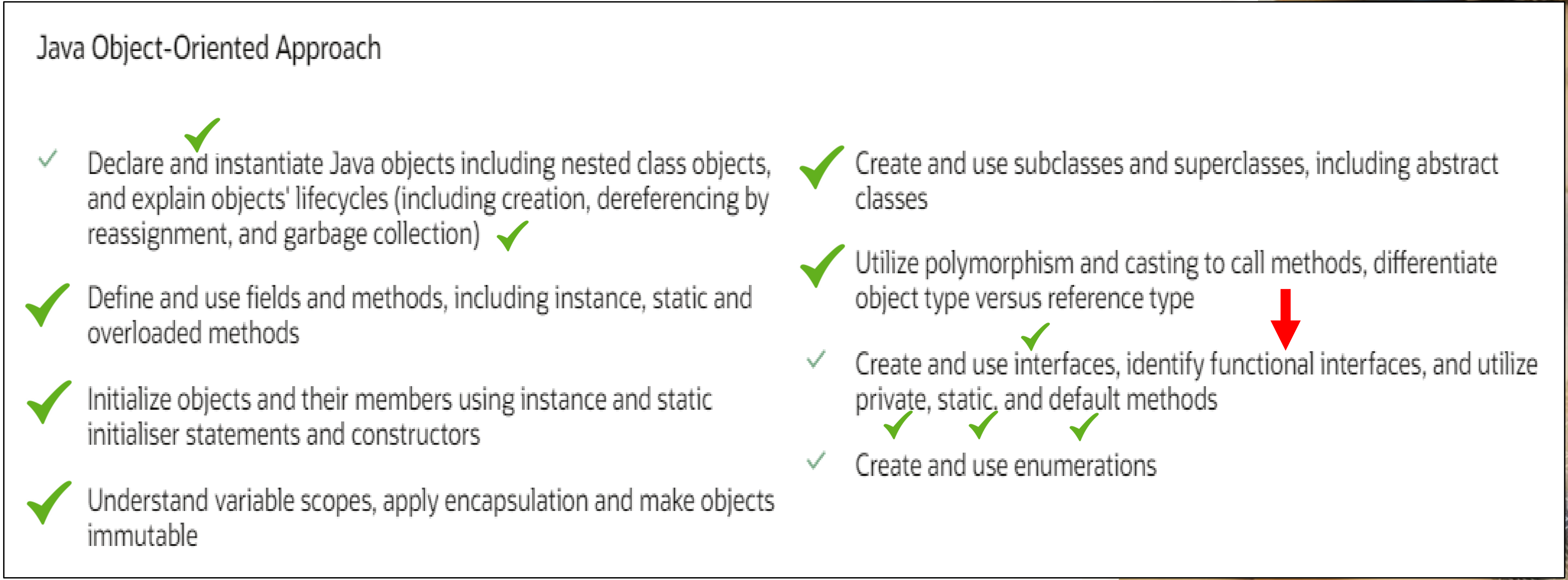
Java Object Oriented Approach

Functional Interfaces

Java Object-Oriented Approach

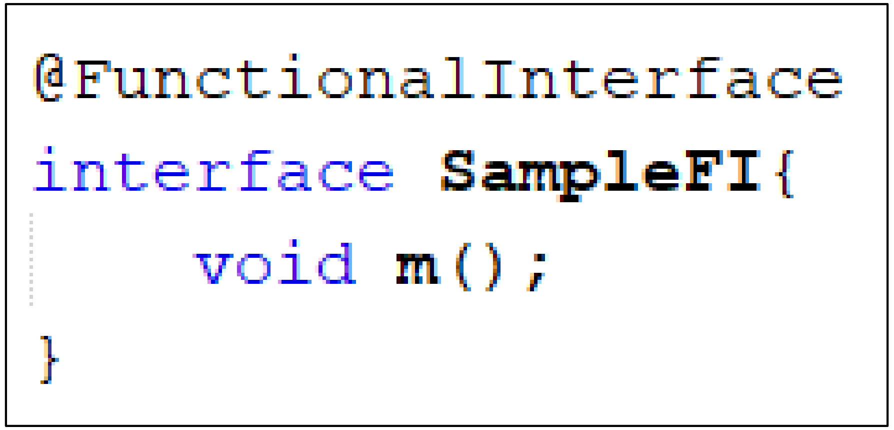


Java Object-Oriented Approach

- ✓ Declare and instantiate Java objects including nested class objects, and explain objects' lifecycles (including creation, dereferencing by reassignment, and garbage collection) ✓
- ✓ Define and use fields and methods, including instance, static and overloaded methods
- ✓ Initialize objects and their members using instance and static initialiser statements and constructors
- ✓ Understand variable scopes, apply encapsulation and make objects immutable
- ✓ Create and use subclasses and superclasses, including abstract classes
- ✓ Utilize polymorphism and casting to call methods, differentiate object type versus reference type
- ✓ Create and use interfaces, identify functional interfaces, and utilize private, static, and default methods
- ✓ Create and use enumerations

Functional Interfaces

- A functional interface is an interface that has **only one abstract** method. This is known as the SAM (Single Abstract Method) rule.
  - *default* methods do not count
  - *static* methods do not count
  - methods inherited from *Object* do not count*



```
@FunctionalInterface
interface SampleFI{
    void m();
}
```