Java Fundamentals Part 2 - Day 1

What is an object?

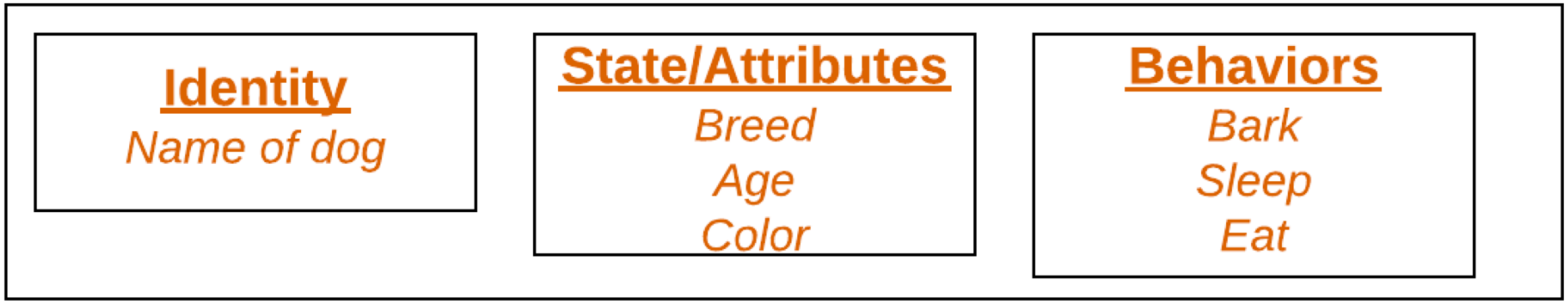
- Anything that may be perceived by the senses
- It can be described by its **attributes** and perform certain **actions**

… and in Java?

- properties in the form of data
- perform certain actions in the form of methods.

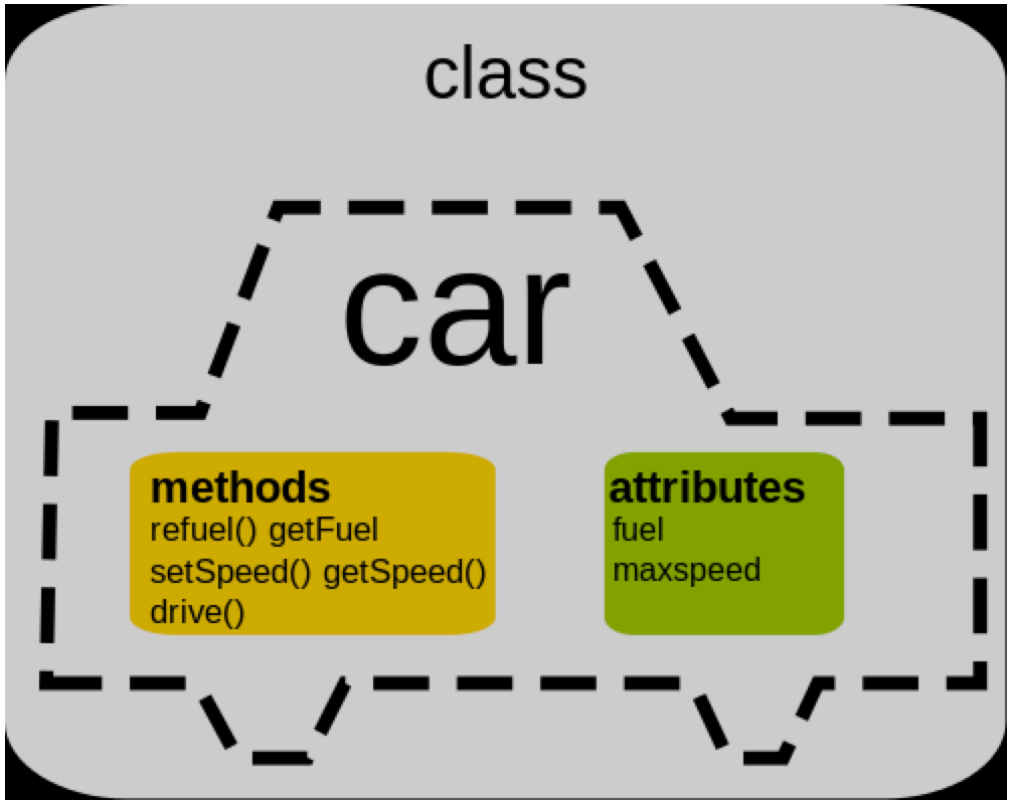
Objects characteristics

- State
- Identity
- Behavior



Class

- A class is a blueprint to create objects that share common properties and methods.



Structure

1. `public` or `default` modifier
2. `class` keyword

3. name
4. superclass ¬ there can be only one.
5. interfaces (if any)
6. Body `{ }`.

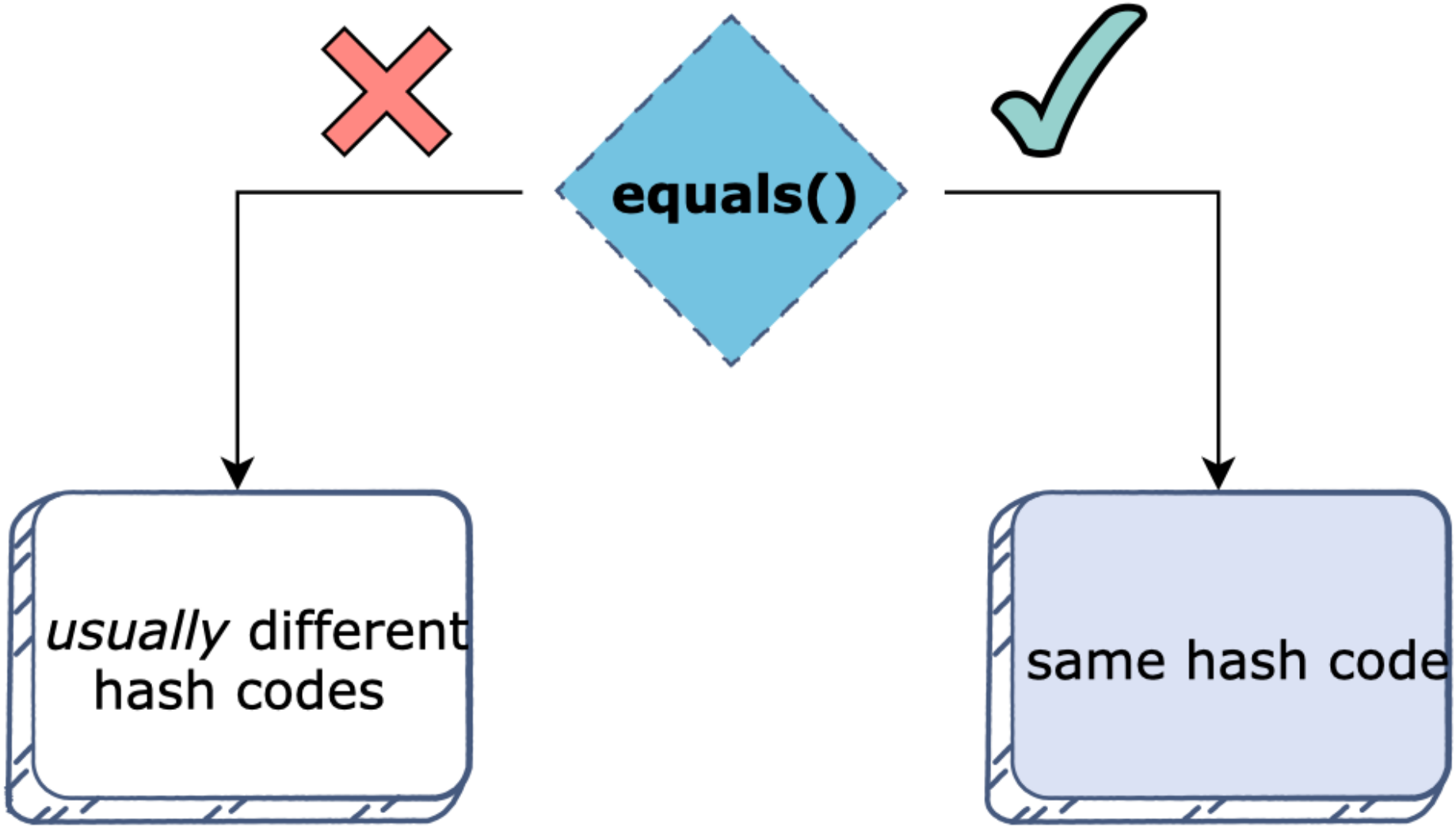
Object class

- parent of all classes
- direct or indirect
- `java.lang`

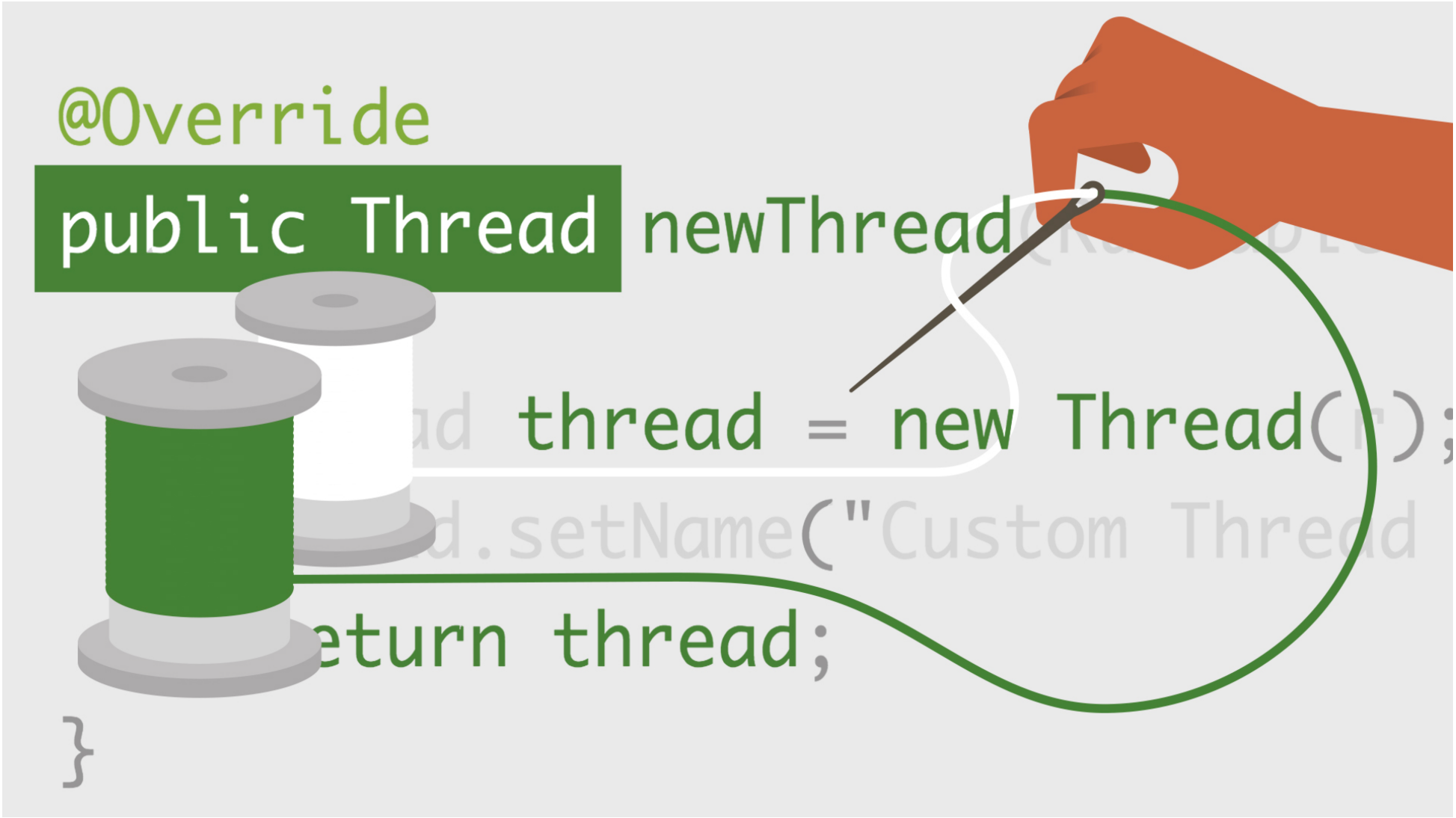
Object Class Methods

- `toString()`
- `hashCode()`
- `equals(Object obj)`
- `getClass()`
- `finalize()`
- `clone()`

equals vs hashCode


equals()
usually different hash codes
same hash code

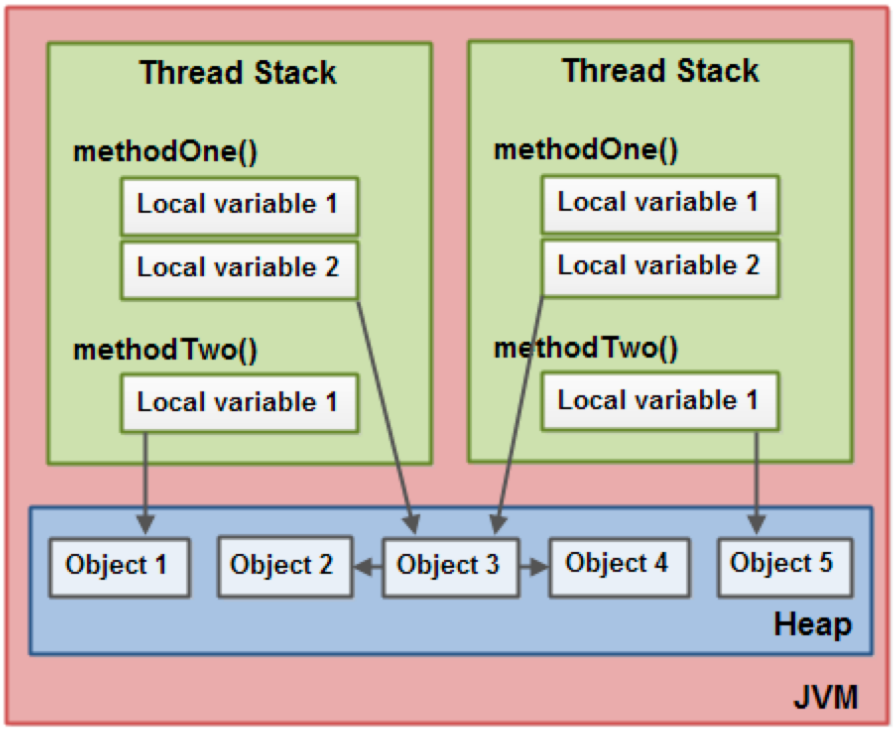
and concurrency-related methods



...

- `wait()`
- `notify()`
- `notifyAll()`

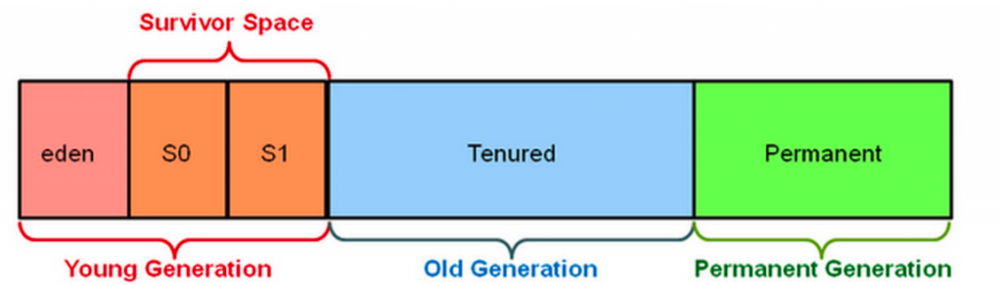
How are Java objects stored in memory?



…

- objects on **heap**
- primitive data types, temporary variables, object addresses on **stack**

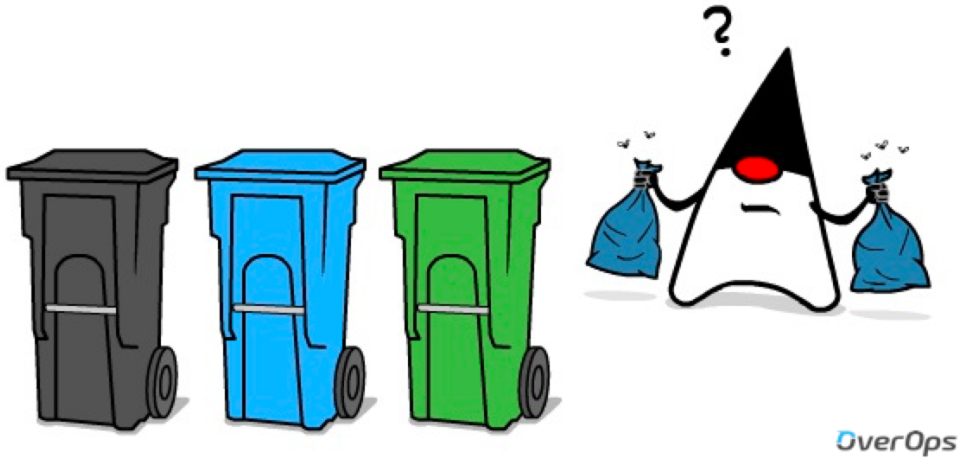
Heap



Stack

- static memory allocation and
- execution of a thread
- primitive values
- references to objects

Garbage Collection



...

- automatic
- frees heap memory
- deletes unused objects

Types ofgarbage collection activity

- Minor or incremental Garbage Collection
- Major or Full Garbage Collection

Unreachable objects

Eligibility for garbage collection

1. Nullifying the reference variable
2. Re-assigning the reference variable
3. An object created inside the method
4. Island of Isolation

Ways to request to run Garbage Collector

1. System.gc()
2. Runtime.getRuntime().gc()

Why we need Garbage Collection?

- makes java memory-efficient
- automatically done

Interfaces

- an abstract type used to specify the behavior of a class
- a blueprint of a class

Why we need interfaces?

- abstraction
- static constants and abstract methods
- sort of multiple inheritance
- loose coupling

Syntax

```
interface Vehicle {

    // the abstract methods.
    void speedUp(int a);

}

class Bicycle implements Vehicle {
    int speed;

    @Override
    public void speedUp(int increment){
        speed = speed + increment;
    }
}
```

What are differences between class and interface?

Abstract class

- a superclass
- declared using the keyword `abstract`.
- an instance cannot be created.
- constructors are allowed.

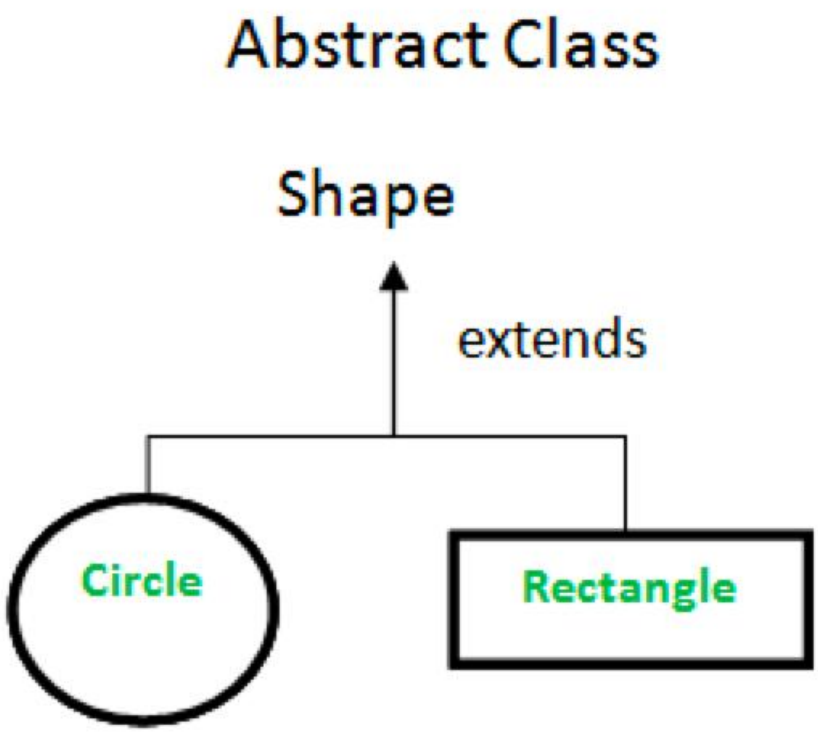
- can have no abstract methods
- can not have final methods
- not allowed to create object for any abstract class
- can define static methods

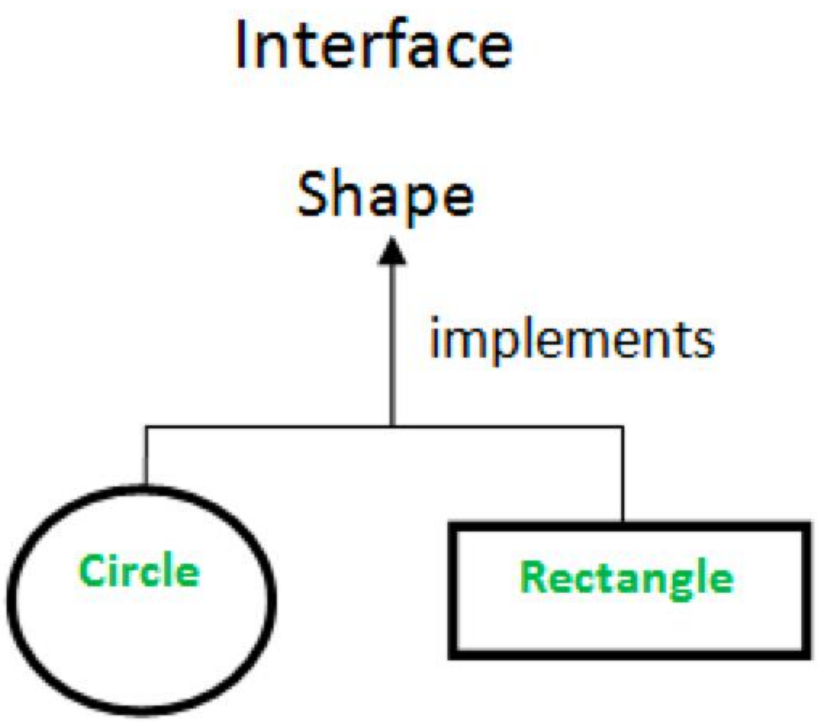
Syntax

```
abstract class Shape {

    abstract void draw();
}

class Circle extends Shape {

    int color;

    void draw(){
        System.out.println("Circle has been drawn");
    };
}
```

What are differences between class and interface?

Abstract Class vs Interface





…

- Type of methods
- Final Variables
- Type of variables
- Implementation
- Inheritance vs Abstraction

- Multiple implementations
- Accessibility of Data Members

When to use what?

Abstract classes

- related classes need to share code
- access ot the non-static or non-final field(s) via a method you can access and modify their state
- classes that extend an abstract class have many common methods or fields

Interfaces

- total abstraction
- multiple inheritance
- specify the behavior but not concerned about who implements it.