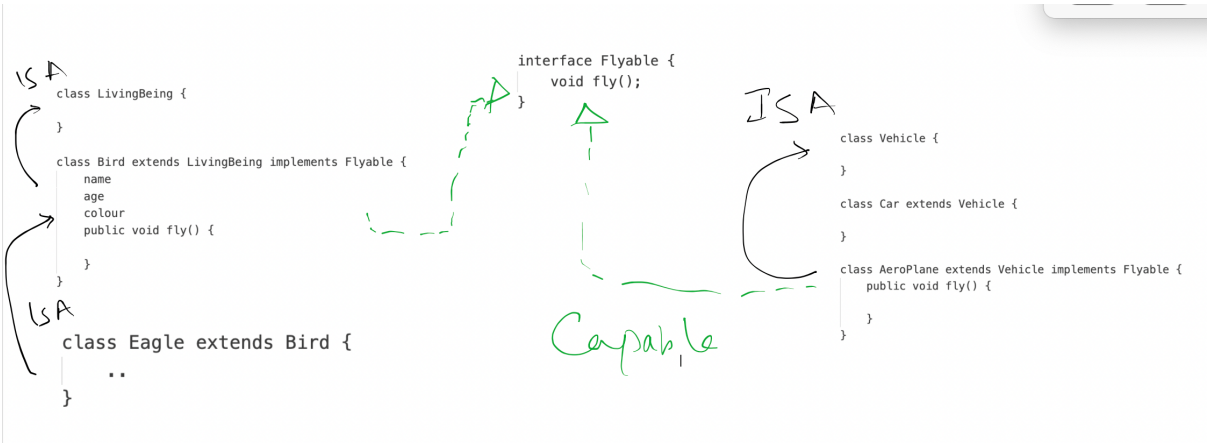



```
interface Dance {
    void dance();
}

interface Fight {
    void fight();
}

interface Swim {
    void swim();
}
```

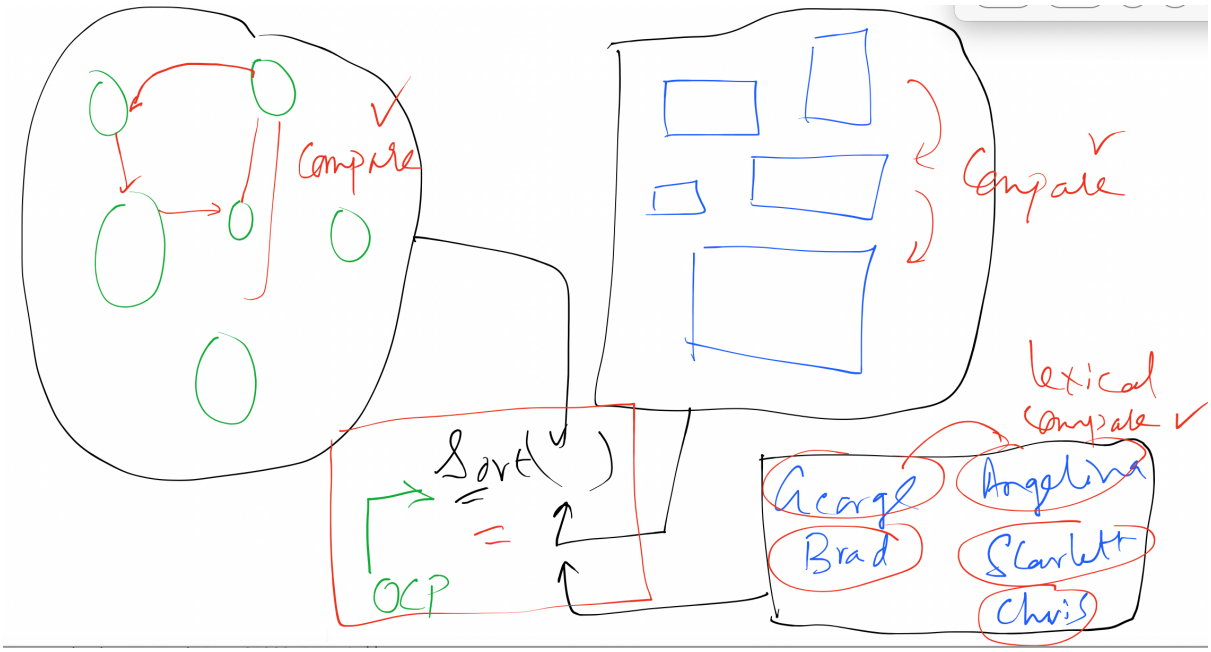
```
public class Actor implements Dance {
    state and behaviour;

    public void dance() {
        ...
    }
}

public class Hero extends Actor implements Fight, Swim {
    ..
    public void fight() {
        ..
    }

    public void swim() {
        ...
    }
}
```

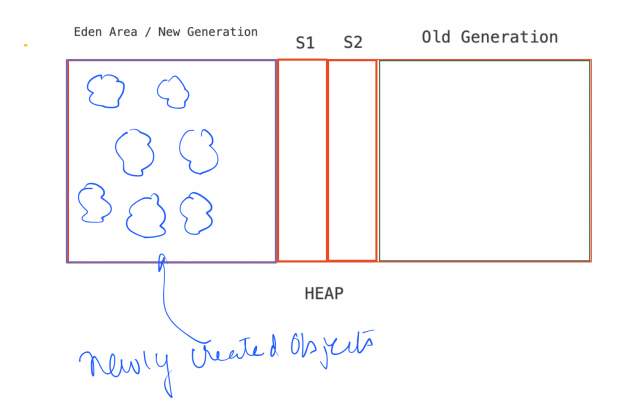
Actor is capable to dance

Hero is a Actor, every actor is capable to dance
hero is also capable to fight and swim

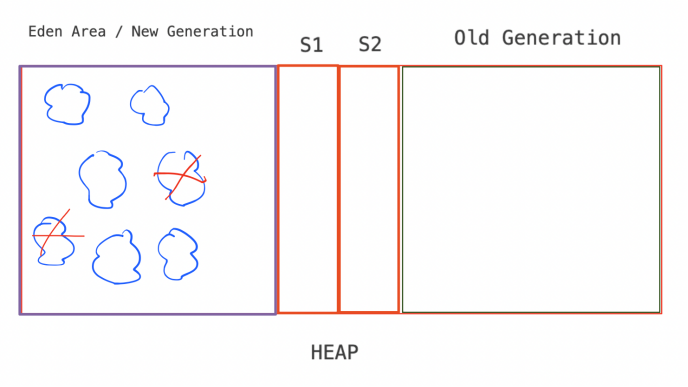
```
Fight f = new Fight(); // not valid; can't instantiate interface, just like abstract class
```

```
// fight interface is referencing a Hero object [stunt master]
Fight f = new Hero(); // valid
f.fight();

f.dance(); // not valid; stunt master can't do choerography

Dance d = (Dance) f;
d.dance(); // valid; hero is now dancing
d.fight(); // not valid

Swim s = (Swim) d; // or f
s.swim(); // valid; hero swims
```

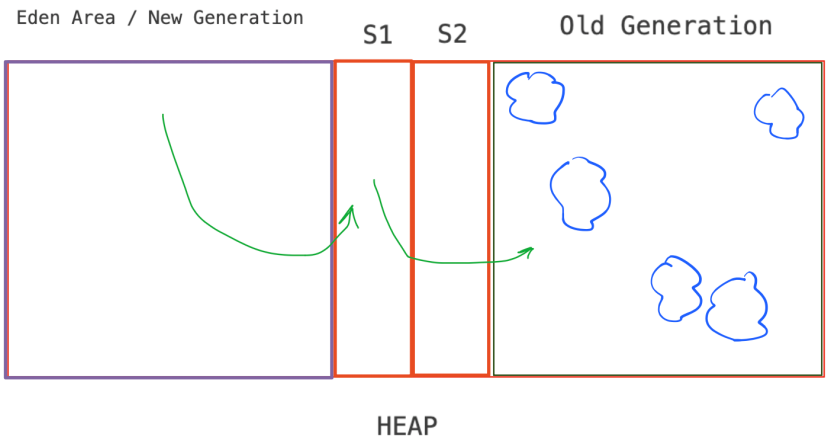


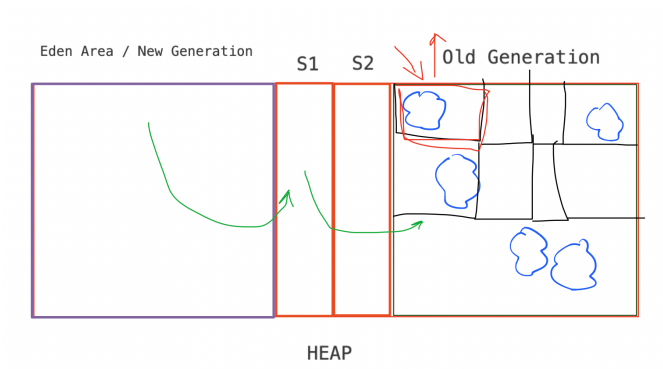


Sort term GC



If objects in Eden area survive 3 cycles of short term GC they are moved to Old Generation by making use of S1 /S2




Eden Area / New Generation
S1
S2
Old Generation
HEAP