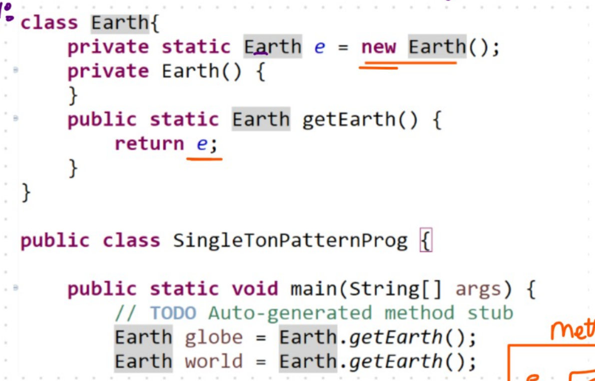
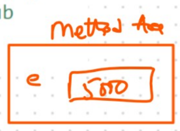
Singleton Pattern

def: In this design pattern, only one object will be created for a class & same Object will be refered by multiple variables

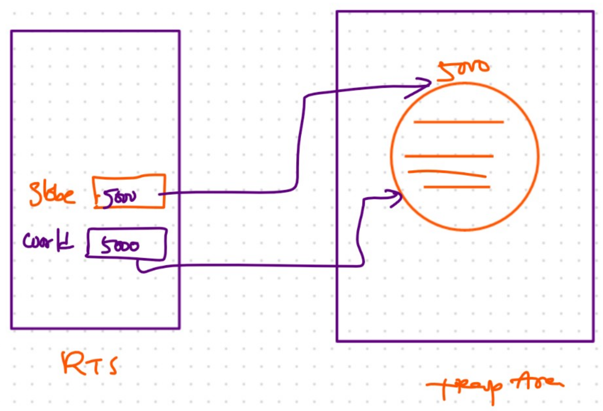
Case1:



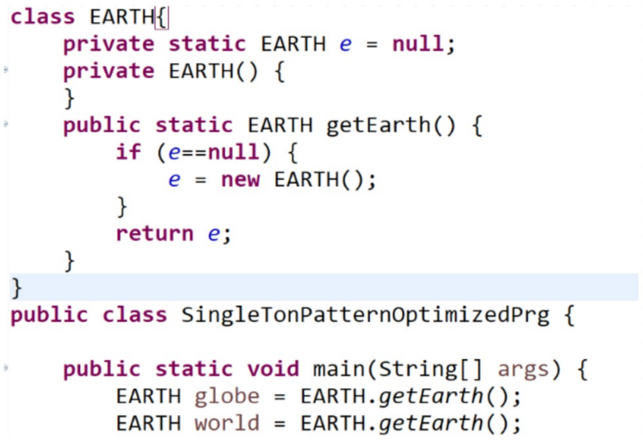
```
class Earth{
    private static Earth e = new Earth();
    private Earth() {
    }
    public static Earth getEarth() {
        return e;
    }
}

public class SingleTonPatternProg {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Earth globe = Earth.getEarth();
        Earth world = Earth.getEarth();
```



Memory Mapping



Case 2



```
class EARTH{
    private static EARTH e = null;
    private EARTH() {
    }
    public static EARTH getEarth() {
        if (e==null) {
            e = new EARTH();
        }
        return e;
    }
}
public class SingleTonPatternOptimizedPrg {

    public static void main(String[] args) {
        EARTH globe = EARTH.getEarth();
        EARTH world = EARTH.getEarth();
```

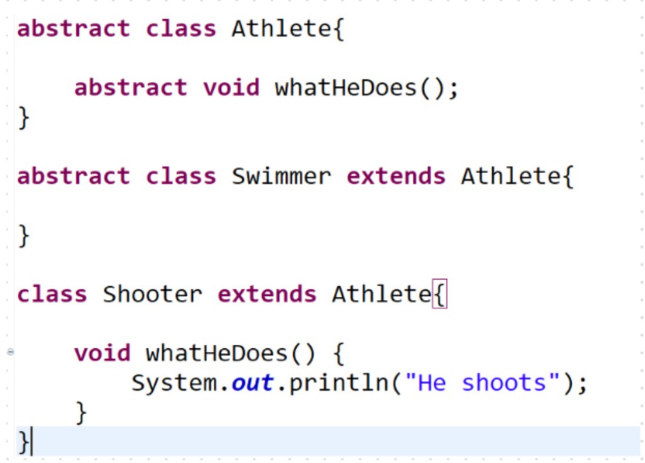
It saves memory if user does not wants any instance of class Earth

Advantages

→ Efficient usage of memory

Abstract

— OOP continuation



```
abstract class Athlete{

    abstract void whatHeDoes();
}

abstract class Swimmer extends Athlete{

}

class Shooter extends Athlete{

    void whatHeDoes() {
        System.out.println("He shoots");
    }
}
```

Notes

→ Abstract is a keyword in Java, which can be used for methods & classes

→ Abstract method does not have body

→ If there is any abstract method in a class, then it is compulsory to make that class as abstract

→ It is not compulsory to have abstract methods inside the abstract class

→ when a subclass [child] inherit from a abstract class, it is compulsory to override the abstract method, inherited from the super class [parent]

(or) alternate ↓

→ we can make the subclass as abstract

→ We cannot create object of abstract class, if we try to create we get an error called " Cannot instantiate the type class ? "

→ we can create the reference of abstract class

ex Athlete shooter = new Shooter()

→ we can have Concrete methods also in abstract class

→ we can have constructors for abstract class

```
abstract class Athlete {
    public Athlete {
        S.o.p ("constructor called");
    }
```

Note: the need of constructor in abstract class is for "constructor chaining"

→ we can have "final methods" inside abstract class

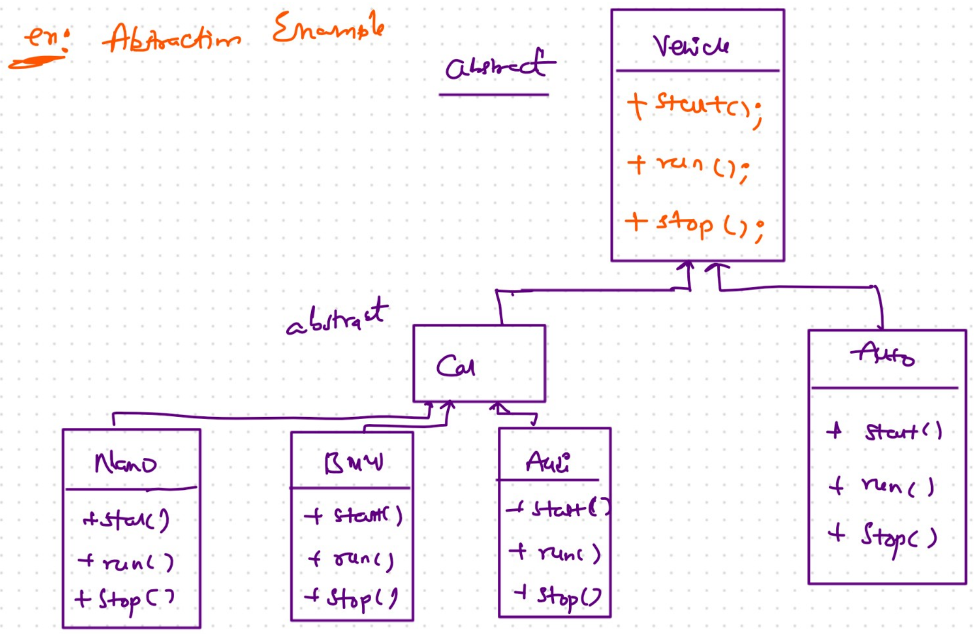
```
abstract class Athlete {
    abstract void whatHeDoes();
    final void sleep() {
        sop ("sleeping")
    }
```

→ we cannot use private & final, static modifier along with abstract keyword

```
abstract static void whatHeDoes();
abstract final void whatHeDoes();

private abstract void whatHeDoes;
```

↓ It doesn't allows you to inherit because we need to define body for abstract method to override

ex: Abstraction Example


abstract
Vehicle
+ start();
+ run();
+ stop();
abstract
Car
Auto
+ start()
+ run()
+ Stop()
Nano
+start()
+run()
+Stop()
BMW
+ start()
+ run()
+ stop()
Audi
+ start()
+ run()
+ stop()