# SPRING

# INVERSION OF CONTROL

**Inversion of Control** (IoC) = mean when someone create object for you . So, instead of writing

[A a= new A()] in your class the object A creates someone else

**Lifecycle** includes:

* creating objects
* destroying them
* invoking methods of the objects of its lifecycle. These methods are called **callbacks**.

IoC is a general concept. DI is one of implementations of IoC.

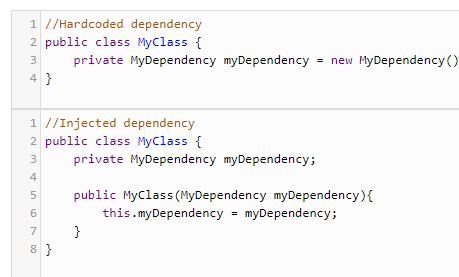
**Ioc can have a few implementations:**

1. using a factory pattern
2. using a service locator pattern
3. using a dependency injection of any given below type:

* a constructor injection
* a setter injection
* an interface injection

# DEPENDENCY INJECTION

**Dependency Injection** – is the opposite to have hardcoded dependencies

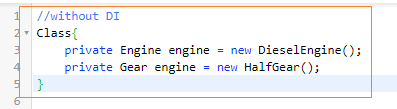


DI can be injected through different ways:

* Constructor
* Set method
* Interface

Example of DI:

There is a class Car. Car consists of wheels, engine and so on



If you need to change config of your object, for example, change type of engine you will need to recreate your Car object. When you use DI objects are given at runtime rather than compile time

DI resolves problems

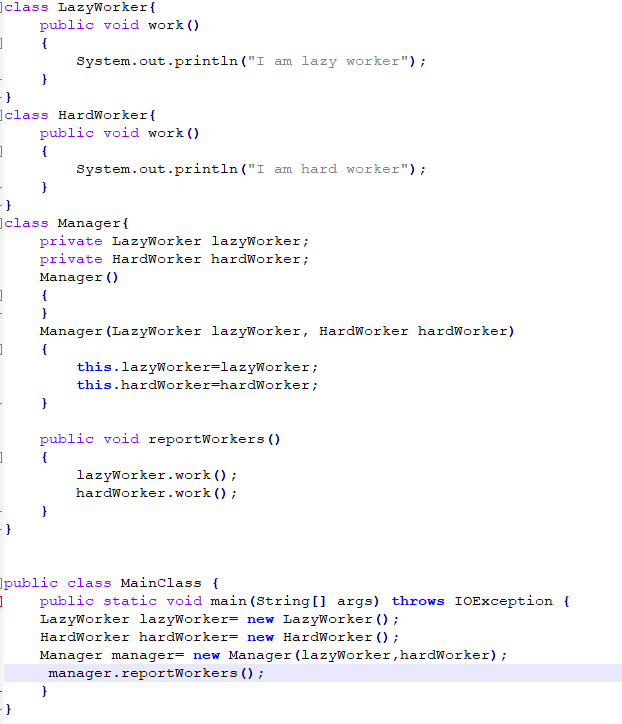
* Reach out loose coupling
* More easy to test jobs
* It satisfies *[open closed principle] of SOLID*

# LOOSE COUPLING

* **Tight coupling** - means classes and object are dependent on another. Tight coupling will be implemented using concrete implementation of classes
* **Loose coupling** - means reducing dependencies of classes that use different classes directly. Loose coupling will be used using classes and interface

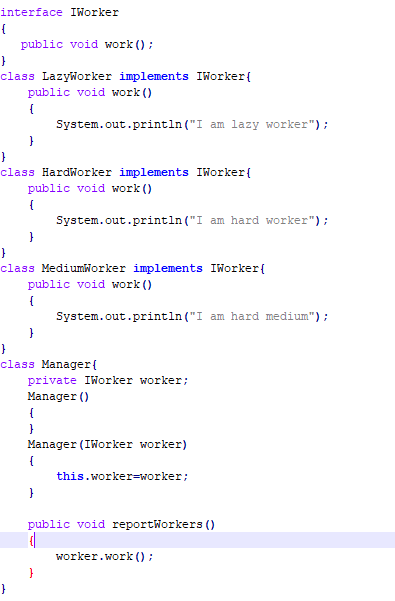
**TIGHT COUPLING EXAMPLE**

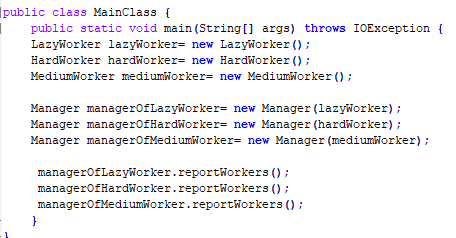
Everything works fine. The problem starts when you need to add one more class. In this case you need to change you constructor of class [Manager]. It means the source class is very **tight**. *It’s not flexible and violate [open closed principle] of SOLID*



**LOOSE COUPLING EXAMPLE**

We use Interface. It means we can add any number of workers and we will not need to change main class [Manager]





**Bean** is the object whose life-cycle managed by the Spring IoC

**Bean** is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

**Spring IoC container** is the central management system of the Spring Framework. It is responsible to create, configure and manage the life cycle of these objects. The container gets its instructions on what objects to instantiate, configure, and assemble by reading configuration metadata provided in the form of either XML configurations or annotations