1. Создать аннотацию **`@AutoInjectable`**
2. Разобраться(поискать в Google) с классом Properties
3. Создать класс **`Injector`** в котором был бы параметризированный метод **`inject`**, который принимал бы в качестве параметра объект любого класса и, используя механизмы рефлексии осуществлял поиск полей, помеченных этой аннотацией(в качестве типа поля используются некоторый интерфейс), и осуществлял бы инициализацию этих полей экземплярами классов, которые указаны в качестве реализации соответствующего интерфейса в некотором файле настроек(properites)

Пример:

У нас есть несколько интерфейсов и несколько классов, реализующих эти интерфейсы:

```
interface SomeInterface{
  public void doSomething();
}
interface SomeOtherInterface{
  public void doSomeOther();
}
class SomeImpl implements SomeInterface{
 public void doSomething(){ println(“A”);}
}
class OtherImpl implements SomeInterface{
 public void doSomething(){ println(“B”);}
}
class SODoer implements SomeOtherInterface{
 public void doSomething(){ println(“C”);}
}
```

У нас есть класс:

```
class SomeBean{
   @AutoInjectable
   private SomeInterface field1;
   @AutoInjectable
   private SomeOtherInterface field2;

   public void foo(){
        field1.doSomething();
        field2.doSomething();
   }
}
```

Обратите внимание, что инициализация полей field1 и field2 нигде в классе не происходит.

Соответственно, если мы где-то в коде напишем

```
(new SomeBean()).foo;
```

то мы получим exception

Ваша задача написать класс **Injector,** который бы осуществлял внедрение зависимостей в любой объект , который содержит поля, помеченные нашей аннотацией. Т.е. некоторый метод этого класса, принимал бы произвольный объект, исследовал бы существующие в нем поля, и смотрел, аннотированы ли они нужной аннотацией.
Если да, то тогда он бы смотрел тип этого поля и искал бы реализацию в файле properties:
Пример файла:

```
somepackage.SomeInterface=somePackage.SomeImpl
somepackage.SomeOtherInterface
                              =somepackage.SODoer
```

После этого, он создавал бы экземпляр нужного класса и записывал ссылку на этот экземпляр в нужное поле. Т.е. для нашего примера метод **`inject`** должен в поле field1 записать экземпляр класса `SomeImpl,` а в поле `field2` класса `SODoer.`

Теперь, инициализация объектов класса SomeBean будет выглядеть следующим образом:

```
SomeBean sb =
      (new Injector()).inject(new SomeBean());
sb.foo();
```

И теперь никаких ошибок быть не должно, а на экран должно вывестись **AC**

Если же мы поменяем в файле properties строчку

```
somepackage.SomeInterface=somePackage.SomeImpl
```

на

```
somepackage.SomeInterface=somePackage. OtherImpl
```

Должно вывестись **BC**