```java
public interface MobileDao {
    void addMobile(Mobile m);
}
```

```java
public class MobileDaoDbImpl implements MobileDao {

    @Override
    public void addMobile(Mobile m) {
        System.out.println(m.getName() + " stored in RDBMS!!!");
    }
}
```

```java
public class MobileDaoMongoImpl implements MobileDao {

    @Override
    public void addMobile(Mobile m) {
        System.out.println(m.getName() + " stored in MogoDB");
    }
}
```

OR

RDBMS

MongoDB

STRATEGY DESIGN PATTERN

SERVICE
CLIENT

```java
public interface MobileDao {
    void addMobile(Mobile m);
}
```

```java
public class MobileDaoDbImpl implements MobileDao {

    @Override
    public void addMobile(Mobile m) {
        System.out.println(m.getName() + " stored in RDBMS!!!");
    }
}
```

```java
public class MobileDaoMongoImpl implements MobileDao {

    @Override
    public void addMobile(Mobile m) {
        System.out.println(m.getName() + " stored in MogoDB");
    }
}
```

RDBMS

MongoDB

FACTORY

TEST CLIENT

WEB CLIENT

MOBILE CLIENT

DESKTOP CLIENT

---

database.properties ✕

1 MOBILE_DAO=com.adobe.prj.dao.MobileDaoDbImpl

```java
public class MobileDaoFactory {
    static String DAO = "";
    // static blocks gets executed as soon as class is loaded into JVM
    static {
        ResourceBundle res = ResourceBundle.getBundle("database");
        DAO = res.getString("MOBILE_DAO");
    }

    public static MobileDao getMobileDao() {
        try {
            return (MobileDao) Class.forName(DAO).newInstance();
        } catch(Exception ex) {
            ex.printStackTrace();
        }
        return null;
    }

//  public static MobileDao getMobileDao() {
//      return new MobileDaoMongoImpl();
//  }

}
```

CREATE INSTANCE OF LOADED CLASS

loads

com.adobe.prj.dao.MobileDaoDbImpl

into JVM

```
interface Dance {
    dance();
}

interface Swim {
    swim();
}

interface Fight {
    fight();
}
```

Choreographer (handwritten, red)

Knows (handwritten, green)

```
class Actor implements Dance {
    state, behaviour
    dance() {
        //
    }
}
```

INHERITED (handwritten, green)

HERO ISA Actor Knows Dance, Swim, Fight (handwritten, green)

```
class Hero extends Actor implements Swim, Fight {
    fight() {}
    swim() { }
}
```

```
Dance d = new Actor();
d.dance();
```

Stunt Master (handwritten, red)

```
Fight f = new Hero();
f.fight(); ✔
```

```
f.dance(); ✗
```

```
Dance d = (Dance) f;
d.dance();
```