20.01.2022

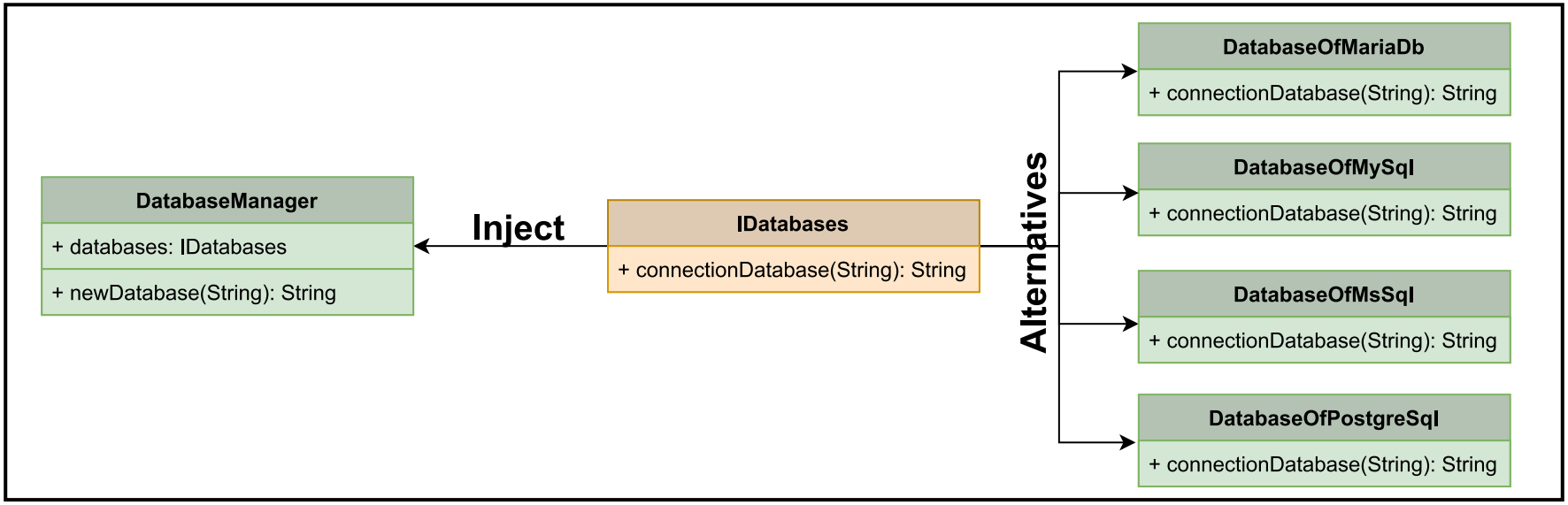
PATİKA – INNOVA JAVA SPRİNG
ODEV – 2

Bekir Gürkan Güldaş

İçindekiler

1. Alternative Annotation

Farklı amaçlar için kullanılan bir bean'nin birden fazla versiyonu mevcut ise bir niteleyiciyi ile enjekte ederek bunlar arasında seçim yapılabilir. Kaynak kodunu değiştirmek zorunda kalmak yerine, alternatifleri kullanarak dağıtım zamanında seçim yapmayı mümkün kılar.



1. Alternative Annotation

```java
package com.innova.CDI;

import java.io.Serializable;

@Named(value = "DatabaseManager")
@ApplicationScoped
public class DatabaseManager implements Serializable{

    private static final long serialVersionUID = 1648478233333869916L;

    @Inject
    private IDatabases databases;

    public String newDatabase()
    {
        return databases.connectionDatabase();
    }
}
```

```java
package com.innova.CDI;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMySql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MySQL.";
    }
}
```

```java
package com.innova.CDI;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMsSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MsSQL.";
    }
}
```

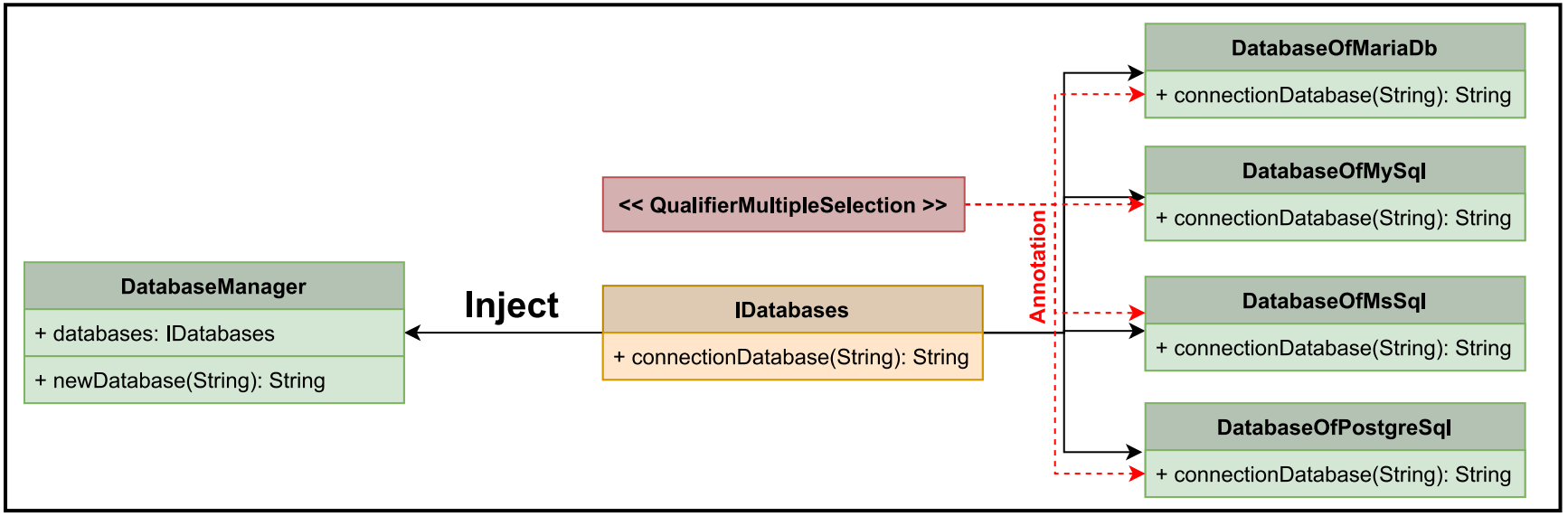
```java
package com.innova.CDI;

public interface IDatabases {

    String connectionDatabase();
}
```

```java
package com.innova.CDI;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfPostgreSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to PostgreSQL.";
    }
}
```

```java
package com.innova.CDI;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMariaDb implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MariaDb.";
    }

}
```

2. Qualifer Annotation

Kullanılan dependency injection konteyner ortamında aynı nesne türünden birden fazla enjekte edilebilir kaynak bulunduğunda, hangisinin seçileceğini belirlemek için kullanılan bir annotationdır.



2. Qualifer Annotation

```java
package com.innova.Qualifier;

import java.io.Serializable;

@Named(value = "QualifierDatabaseManager")
@ApplicationScoped
public class DatabaseManager implements Serializable{

    private static final long serialVersionUID = 16484782333338699916L;

    //Default Inject

    //@Inject
    //private IDatabases databases;

    //Qualifier Inject

    @Inject
    @QualifierMultipleSelection
    private IDatabases databases;

    public String newDatabase()
    {
        return databases.connectionDatabase();
    }
}
```

```java
package com.innova.Qualifier;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.inject.Qualifier;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

@Qualifier
@Target({ TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
public @interface QualifierMultipleSelection {

}
```

2. Qualifer Annotation

```java
package com.innova.Qualifier;

import javax.enterprise.inject.Default;

@Default
public class DatabaseOfMariaDb implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MariaDb.";
    }

}
```

```java
package com.innova.Qualifier;

public class DatabaseOfPostgreSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to PostgreSQL.";
    }
}
```

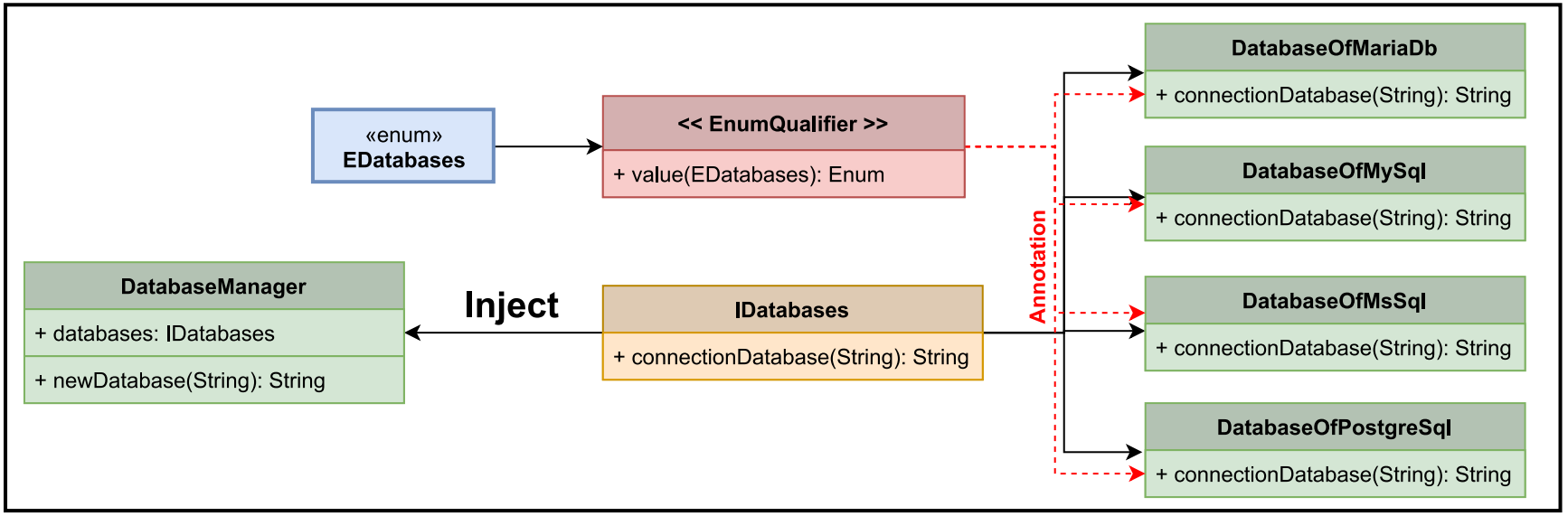
```java
package com.innova.Qualifier;

public class DatabaseOfMySql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MySQL.";
    }
}
```

```java
package com.innova.Qualifier;

public interface IDatabases {

    String connectionDatabase();
```

```java
package com.innova.Qualifier;

@QualifierMultipleSelection
public class DatabaseOfMsSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MsSQL.";
    }
}
```

3. Enumqualifer Annotation

Enjekte edilebilir kaynakların tek bir notasyonla seçilebilir kılmak için aşağıdaki gibi bir enum oluşturulabilir.



3. Enumqualifer Annotation

```java
package com.innova.EnumQualifier;

import java.io.Serializable;

import javax.enterprise.context.ApplicationScoped;
import javax.inject.Inject;
import javax.inject.Named;

@Named(value = "EnumQualifierDatabaseManager")
@ApplicationScoped
public class DatabaseManager implements Serializable{

    private static final long serialVersionUID = 1648478233333869916L;

    //Default Inject

    //@Inject
    //private IDatabases databases;

    //Enum Qualifier Inject

    @Inject
    @EnumQualifier(EDatabases.MSSQL)
    private IDatabases databases;

    public String newDatabase()
    {
        return databases.connectionDatabase();
    }

}
```

```java
package com.innova.EnumQualifier;

public enum EDatabases
{
    MariaDB, MSSQL, MySQL, PostgreSQL
}
```

```java
package com.innova.EnumQualifier;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.PARAMETER;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.inject.Qualifier;

@Qualifier
@Target({ TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
public @interface EnumQualifier {
    EDatabases value();
}
```

3. Enumqualifer Annotation

```java
package com.innova.EnumQualifier;

import javax.enterprise.inject.Default;

@Default
public class DatabaseOfPostgreSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to PostgreSQL.";
    }
}
```

```java
package com.innova.EnumQualifier;

@EnumQualifier(EDatabases.MySQL)
public class DatabaseOfMySql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MySQL.";
    }
}
```

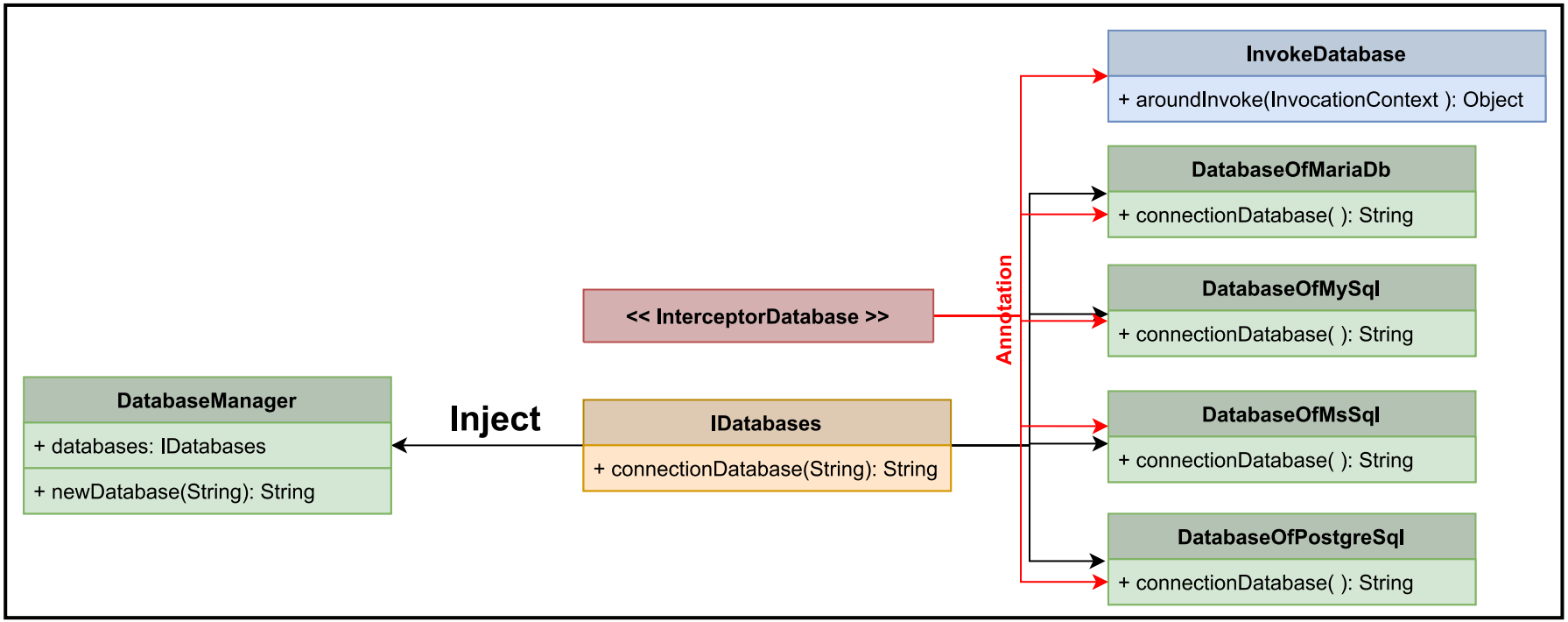
```java
package com.innova.EnumQualifier;

public interface IDatabases
{
    String connectionDatabase();
}
```

```java
package com.innova.EnumQualifier;

@EnumQualifier(EDatabases.MariaDB)
public class DatabaseOfMariaDb implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MariaDb.";
    }
}
```

```java
package com.innova.EnumQualifier;

@EnumQualifier(EDatabases.MSSQL)
public class DatabaseOfMsSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MsSQL.";
    }
}
```

4. Interceptor Annotation

Yazılım geliştirme alanında bir önleyici desendir. Yazılım sistemleri veya çerçeveleri normal işlem döngülerini değiştirmek veya artırmak için bir yol sunmak istediğinde kullanılan bir yazılım tasarım modelidir.



4. Interceptor Annotation

```java
package com.innova.Interceptor;

import java.io.Serializable;

import javax.enterprise.context.ApplicationScoped;
import javax.inject.Inject;
import javax.inject.Named;

@Named(value = "InterceptorDatabaseManager")
@ApplicationScoped
public class DatabaseManager implements Serializable{

    private static final long serialVersionUID = 16484782333333869916L;

    @Inject
    private IDatabases databases;

    public String newDatabase()
    {
        return databases.connectionDatabase();
    }

}
```

```java
package com.innova.Interceptor;

import java.lang.annotation.Documented;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.interceptor.InterceptorBinding;

import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

@InterceptorBinding
@Inherited
@Target({ TYPE, METHOD })
@Retention(RUNTIME)
@Documented
public @interface InterceptorDatabase {

}
```

```java
package com.innova.Interceptor;

import javax.interceptor.AroundInvoke;
import javax.interceptor.Interceptor;
import javax.interceptor.InvocationContext;

@Interceptor
@InterceptorDatabase
public class InvokeDatabase {

    @AroundInvoke
    public Object aroundInvoke(InvocationContext context) {

        boolean isLogin = false;
        Object isContinue = null;
        if (isLogin)
        {
            System.out.println("Connection Failed.");
            return null;
        }
        else
        {
            try
            {
                isContinue = context.proceed();
                System.out.println("Connection Successed.");
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
        return isContinue;
    }
}
```

4. Interceptor Annotation

```java
package com.innova.Interceptor;

import javax.enterprise.inject.Alternative;

@InterceptorDatabase
@Alternative
public class DatabaseOfPostgreSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to PostgreSQL.";
    }
}
```

```java
package com.innova.Interceptor;

import javax.enterprise.inject.Alternative;

@InterceptorDatabase
@Alternative
public class DatabaseOfMySql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MySQL.";
    }
}
```

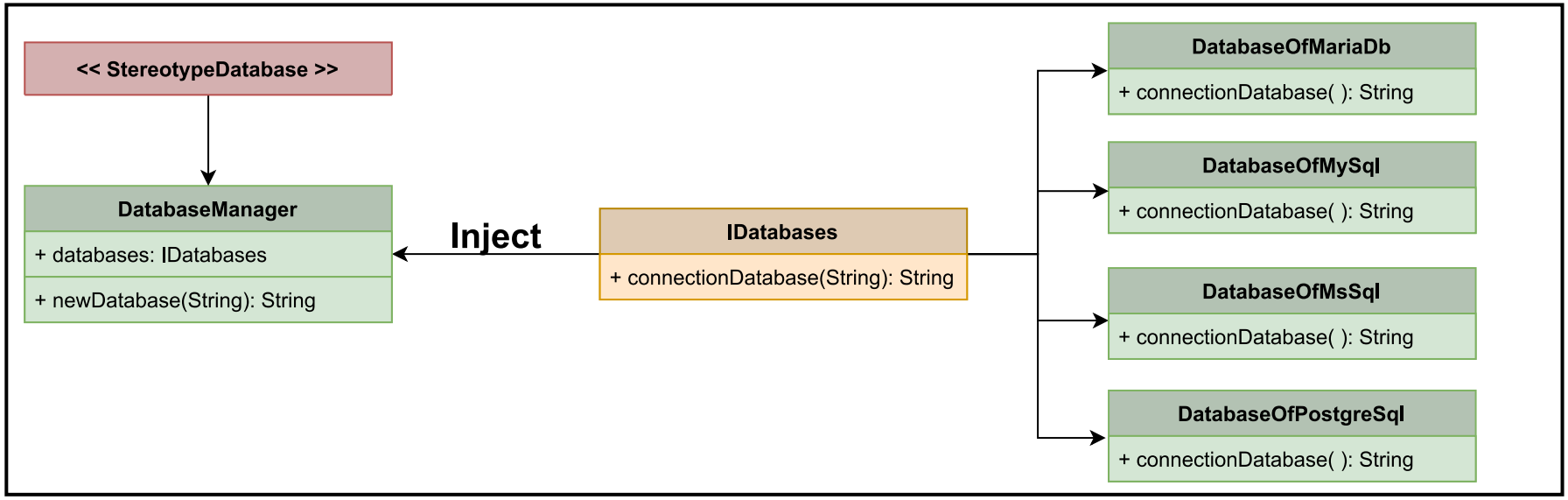
```java
package com.innova.Interceptor;

public interface IDatabases
{
    String connectionDatabase();
}
```

```java
package com.innova.Interceptor;

import javax.enterprise.inject.Alternative;

@InterceptorDatabase
@Alternative
public class DatabaseOfMsSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MsSQL.";
    }
}
```

```java
package com.innova.Interceptor;

import javax.enterprise.inject.Alternative;

@InterceptorDatabase
@Alternative
public class DatabaseOfMariaDb implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MariaDb.";
    }

}
```

5. Streotypes Annotation

Stereotype anotasyonu, bir uygulama içinde bir rolü yerine getiren herhangi bir sınıf için belirteçlerdir. Bileşenler için gereken yapılandırmaların kaldırılmasına veya en azından büyük ölçüde azaltılmasına yardımcı olur.



5. Streotypes Annotation

```java
package com.innova.Stereotype;

import java.io.Serializable;
import javax.inject.Inject;

@StereotypeDatabase
public class DatabaseManager implements Serializable{

    private static final long serialVersionUID = 1648478233333869916L;

    @Inject
    private IDatabases databases;

    public String newDatabase()
    {
        return databases.connectionDatabase();
    }
}
```

```java
package com.innova.Stereotype;

import static java.lang.annotation.ElementType.FIELD;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;

import javax.enterprise.context.ApplicationScoped;
import javax.enterprise.inject.Stereotype;
import javax.inject.Named;

@Stereotype
@Target({ TYPE, METHOD, FIELD })
@Retention(RUNTIME)
@Documented

@Named
@ApplicationScoped
public @interface StereotypeDatabase {

}
```

5. Streotypes Annotation

```java
package com.innova.Stereotype;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMySql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MySQL.";
    }
}
```

```java
package com.innova.Stereotype;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfPostgreSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to PostgreSQL.";
    }
}
```

```java
package com.innova.Stereotype;

public interface IDatabases
{
    String connectionDatabase();
}
```

```java
package com.innova.Stereotype;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMsSql implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MsSQL.";
    }
}
```

```java
package com.innova.Stereotype;

import javax.enterprise.inject.Alternative;

@Alternative
public class DatabaseOfMariaDb implements IDatabases{

    @Override
    public String connectionDatabase()
    {
        return "Connected to MariaDb.";
    }

}
```

Kaynak Kod : https://github.com/gurkanguldas/InnovaCdiProject