JAVA BASICS

public class Main {  
  
 public static void main(String[] args) {  
 // Array tanımlamak, döngüler, şart koşulları, değişken tanımlamaları vs. c# ile aynıdır.  
 String[] ogrenciler = new String[4];  
 ogrenciler[0] = "Furkan";  
 ogrenciler[1] = "Keskin";  
 ogrenciler[2] = "Sen Bir";  
 ogrenciler[3] = "Keksin";  
  
 // for döngüsü c#'daki gibi kullanım ile de çalışır.  
 for (String ogrenci : ogrenciler) {  
 // Konsola yazı yazar.  
 System.out.println(ogrenci);  
 }  
  
 String mesaj = "Bugün hava çok sıcak.";  
  
 // String'in eleman sayısını verir.  
 System.out.println("Eleman sayısı: " + mesaj.length());  
  
 // String'in istenen index'indeki elemanını verir.  
 System.out.println("5. eleman: " + mesaj.charAt(4));  
  
 // String'e ekleme yapar.  
 System.out.println(mesaj.concat(" Eklenen cümle."));  
  
 // String'in istenen harfle başlayıp başlamadığına bakar. (bool)  
 System.out.println(mesaj.startsWith("A"));  
  
 // String'in istenen harfle bitip bitmediğine bakar. (bool)  
 System.out.println(mesaj.endsWith("."));  
  
 // String'in istenen harflerini char dizisine aktarma.  
 char[] karakterler= new char[5];  
 // mesaj'ın 5-10 arasındaki karakterlerini karakterler dizisine 0. indexten itibaren ata.  
 mesaj.getChars(5,10,karakterler,0);  
 System.out.println(karakterler);  
  
 // String'de istenen karakterlerin indexinin başladığı indexi öğrenme.  
 System.out.println(mesaj.indexOf("hava"));  
  
 // String'de istenen karakterlerin indexinin başladığı yeri sondan arayarak bulur.  
 // İki fonksiyon da ilk bulduğu indexi verip durur.  
 // Aranan karakterler sadece 1 kere varsa sonuç değişmez. İndexi hep soldan sayar.  
 System.out.println(mesaj.lastIndexOf("hava"));  
 // replace, toLowerCase, toUpperCase, split, substring gibi fonksiyonlar da var.  
 // trim fonksiyonu ile string'in başındaki ve sonundaki gereksiz boşluklar silinir.  
 }  
}

METHODS

public class Main {  
  
 // Methodlar c# ile aynı sayılır.  
 public static void main(String[] args) {  
 System.out.println(topla(5,3,5));  
 }  
  
 // Variable Arguments  
 // ... paramsa karşılık gelir. sayilari array'e çevirir.  
 public static int topla(int... sayilar){  
 int toplam =0;  
 for(int sayi:sayilar){  
 toplam+=sayi;  
  
 }  
 return toplam;  
 }  
}

CLASSES

CustomerManager.class

public class CustomerManager {  
 public void Add(){  
 System.out.println("Müşteri Eklendi");  
 }  
 public void Delete(){  
 System.out.println("Müşteri Silindi");  
 }  
 public void Update(){  
 System.out.println("Müşteri Güncellendi");  
 }  
}

Main.class

public class Main {  
  
 public static void main(String[] args) {  
 CustomerManager customerManager=new CustomerManager();  
 customerManager.Add();  
 customerManager.Delete();  
 customerManager.Update();  
 }  
}

ENCAPSULATION

Product.class

public class Product {  
  
 // C#'daki propertylerle aynı görevi görür.  
 // attribute | field  
 // C#'da değişkenler otomatik private olurken, java'da public oluyor.  
 private int \_id;  
 private String \_name;  
 private String \_description;  
 private double \_price;  
 private int \_stockAmount;  
  
 //Constructor  
 public Product(){  
  
 }  
  
 //Constructor  
 public Product(int id, String name, String description, double price, int stockAmount){  
 System.out.println("Constructor çalıştı");  
 \_id=id;  
 \_name=name;  
 \_description=description;  
 \_price=price;  
 \_stockAmount=stockAmount;  
  
 }  
  
 // get, set yazımı.  
 // get okumaya, set değer vermeye yarar. Buna göre kullanılabilirsin.  
 // Değişkenleri seçip, sağ tık refactor encapsulete all denip yapılabilir.  
 public int getId(){  
 return \_id;  
 }  
  
 public void setId(int id){  
 \_id=id;  
 }  
  
 public String getName() {  
 return \_name;  
 }  
 // this classın en üstündeki \_name'i kast eder.  
 public void setName(String name) {  
 this.\_name = name;  
 }  
  
 public String getDescription() {  
 return \_description;  
 }  
  
 public void setDescription(String description) {  
 this.\_description = description;  
 }  
  
 public double getPrice() {  
 return \_price;  
 }  
  
 public void setPrice(double price) {  
 this.\_price = price;  
 }  
  
 public int getStockAmount() {  
 return \_stockAmount;  
 }  
  
 public void setStockAmount(int stockAmount) {  
 this.\_stockAmount = stockAmount;  
 }  
}

ProductManager.class

public class ProductManager {  
 public void Add(Product product){  
 System.out.println("Ürün eklendi "+product.getName());  
 }  
}

Main.class

public class Main {  
  
 public static void main(String[] args) {  
 // Ancak constructor tanımlarsak bu yöntemi kullanabiliriz.1  
  
 Product product=new Product(1,"Laptop", "Asus Laptop", 5000,3);  
  
 // Overload yapmazsak alttaki satır çalışmaz.2  
 /\*  
 product.setId(1);  
 product.setName("Laptop");  
 product.setDescription("Asus Laptop");  
 product.setPrice(5000);  
 product.setStockAmount(3);\*/  
  
 ProductManager productManager=new ProductManager() ;  
 productManager.Add(product);  
 System.out.println(product.getName());  
 }  
}

INHERITANCE

// extends keyword'u ile inherit yapılır  
public class Employee extends Person{  
 double salary;  
}

// inherit eden classların fonksiyonları override etmemesi için public final void örneği gibi kullanımlar yapılabilir.

ABSTRACT CLASSES

Main.class

public class Main {  
 public static void main(String[] args) {  
 Adult adult=new Adult();  
 adult.hesapla();  
 adult.gameOver();  
  
 }  
}

GameCalculator.class

public abstract class GameCalculator{  
 // abstract classlar newlenirken içeriğinin tekrar yazılması gerekir.  
 // içerisine abstract fonksiyonlar atılabilir.  
 // bu fonksiyonların base'i yoktur. abstract class'ı inherit eden her class kendine özel şekilde doldurmalıdır.  
 public abstract void hesapla();  
  
 // final içeren fonksiyonlar override edilemez.  
 public final void gameOver(){  
 System.out.println("Game Over");  
 }  
}

Adult.class

public class Adult extends GameCalculator {  
 // Bir fonksiyonu override ettiğimizde @Override ile işaretleriz ki anlaşılsın.  
 @Override  
 public void hesapla() {  
 System.out.println("Adult calculator works");  
 }  
}

INTERFACES

// Same as c#  
// Classlar birden fazla interface'i implemente edebilirler.

COMPOSITION, INNER CLASS, STATIC

// Bir fonksiyon static tanımlanırsa her yerden Classİsmi.FonksiyonAdı şeklinde ulaşılabilir.  
// Hafızadan hiç atılmaz. Tek nesne olarak tanımlanır.  
// Böylece class'ı newlemeye ihtiyaç duymayız.  
// Çağırıldığında sadece fonksiyonun içi çalışır.  
// Constructorlar sadece newlenince değişir. C#'da parametresiz constructor static fonksiyon çağırılınca da çalışır.  
// Class içerisine static bir constructor tanımlarsak fonksiyon çalışınca çalışır. Birden fazla oluşturulabilir.  
// C#'da ana class static olabilirken, java'da olamaz.  
// Ancak java'da inner class oluşturulursa static class oluşturulabilir.

PACKAGES

Main.class

// Kullanıcıdan girdi almaya yarayan kütüphane.  
import java.util.Scanner;  
  
import matematik.DortIslem;  
// import matematik.\* yaparak package içerisindeki tüm classları çağırabiliriz.  
  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 // Kullanıcıdan veri alınır.  
 System.out.print("Adınızı Giriniz: ");  
 String mesaj = scanner.nextLine();  
 System.out.println(mesaj);  
  
 // Kullanıcıdan gelen integeri alma.  
 System.out.print("Yaşınızı Giriniz: ");  
 // int sayi = scanner.nextInt();  
 // Tür dönüşümü yapma.  
 int sayi=Integer.parseInt(scanner.nextLine());  
 System.out.println(sayi);  
  
 // Kendi Package'imiz.  
 DortIslem dortIslem=new DortIslem();  
 dortIslem.topla(2,3);  
 }  
}

DortIslem.class

package matematik;  
// src içerisine yeni bir package ekledik.  
// onun içerisine de DortIslem.class'ı ekledik.  
// Kendi paketimizi oluşturarak gruplama yaparız.  
// Import ederek kullanabiliriz.  
  
public class DortIslem {  
 public void topla(int sayi1, int sayi2){  
 int toplam= sayi1+sayi2;  
 System.out.println("Toplam "+toplam);  
 }  
}

COLLECTIONS

import java.util.ArrayList;  
  
public class Main {  
  
 public static void main(String[] args) {  
 ArrayList liste = new ArrayList();  
  
 // Listeye ekleme yapma  
 liste.add(1);  
 liste.add("Kastamonu");  
 liste.add(222);  
  
 // Listenin boyutunu öğrenme  
 System.out.println("Listenin Boyutu: " + liste.size());  
  
 // Listeden eleman çekme  
 System.out.println(liste.get(1));  
  
 // Listeden bir eleman değiştirme.  
 liste.set(1, "İstanbul");  
  
 // Listeden eleman silme  
 liste.remove(2);  
  
 // Listeyi tamamen silme  
 //liste.clear();  
  
 for(Object eleman:liste){  
 System.out.println(eleman);  
 }  
 }  
}

import java.util.ArrayList;  
import java.util.Collections;  
  
public class Main {  
  
 public static void main(String[] args) {  
 ArrayList<String> cities = new ArrayList<String>();  
 cities.add("İstanbul");  
 cities.add("Ankara");  
 cities.add("Kastamonu");  
 cities.add("Ankara");  
  
 // İndex numarasıyla silme yapılabildiği gibi elemanla da silme yapılabilir.  
 // Bulduğu ilk elemanı siler.  
 cities.remove("Ankara");  
  
 // Collections ile sıralanabilir.  
 Collections.sort(cities);  
  
 for (String sehir : cities) {  
 System.out.println(sehir);  
 }  
 }  
}

EXCEPTION HANDLING

public class Main {  
import java.io.BufferedReader;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
  
  
public class Main {  
  
 public static void main(String[] args){  
  
 // FileReader checked exception olduğu için try catch açmak zorunlu.  
 // reader'ı finally'de kapatabilmek için dışarda boş olarak tanımlıyoruz.  
 BufferedReader reader = null;  
  
 int total =0;  
  
 try {  
 reader = new BufferedReader(new FileReader("C:\\Users\\Furkan\\Desktop\\Programming\\JAVA\\exceptionHandlingDemo\\src\\sayilar.txt"));  
  
 String line =null;  
  
 // reader.readLine bir satır oku demek.  
 // satır boş değilse satırı integer'a çevirip, total'e ekle.  
 while((line =reader.readLine())!=null)  
 {  
 total += Integer.valueOf(line);  
 }  
  
 System.out.println("Toplam = " + total);  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 reader.close();  
 } catch (Exception exception) {  
  
 }  
  
 }  
 }  
}

Throw Keyword

Main.class

public class Main {  
  
 public static void main(String[] args) {  
 AccountManager manager=new AccountManager();  
 System.out.println("Hesap = "+manager.getBalance());  
 manager.deposit(100);  
 System.out.println("Hesap = "+manager.getBalance());  
 try{  
 manager.withdraw(90);  
 }catch (BalanceInsufficentException exception){  
 System.out.println(exception.getMessage());  
 }  
  
 System.out.println("Hesap = "+manager.getBalance());  
 try{  
 manager.withdraw(90);  
 }catch (BalanceInsufficentException exception){  
 System.out.println(exception.getMessage());  
 }  
 System.out.println("Hesap = "+manager.getBalance());  
 }  
}

AccountManager.class

public class AccountManager {  
 private double balance;  
  
 public void deposit(double amount){  
 balance = getBalance() + amount;  
 }  
 public void withdraw(double amount) throws BalanceInsufficentException {  
 if(balance>=amount){  
 balance = getBalance() - amount;  
 }else{  
 throw new BalanceInsufficentException("Bakiye yetersiz");  
 }  
  
 }  
 public double getBalance() {  
 return balance;  
 }  
}

BalanceInsufficentException.class

public class BalanceInsufficentException extends Exception {  
 String message;  
 public BalanceInsufficentException(String message){  
 this.message = message;  
 }  
 @Override  
 public String getMessage() {  
 return this.message;  
 }  
}

WORKING WITH FILES

import java.io.\*;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 createFile();  
 getFileInfo();  
 readFile();  
 writeFile();  
 readFile();  
 }

// Dosya yaratma.  
 public static void createFile() {  
 File file = new File("C:\\Users\\Furkan\\Desktop\\Programming\\JAVA\\workingWithFiles\\src\\testyapmakicin.txt");  
  
 try {  
 // Yeni dosya oluşturma  
 if (file.createNewFile()) {  
 System.out.println("Dosya oluşturuldu");  
 } else {  
 System.out.println("Dosya zaten mevcut");  
 }  
 } catch (IOException e) {  
 // Hata mesajını yazdırır.  
 e.printStackTrace();  
 }  
 }

// Dosya bilgilerini alma.  
 public static void getFileInfo() {  
 File file = new File("C:\\Users\\Furkan\\Desktop\\Programming\\JAVA\\workingWithFiles\\src\\testyapmakicin.txt");  
 if (file.exists()) {  
 System.out.println("Dosya Adı: " + file.getName());  
 System.out.println("Dosya Yolu: " + file.getAbsolutePath());  
 System.out.println("Dosya Yazılabilir Mi : " + file.canWrite());  
 System.out.println("Dosya Okunabilir Mi : " + file.canRead());  
 System.out.println("Dosya Boyutu (byte) : " + file.length());  
  
 }  
 }

// Dosya okuma.  
 public static void readFile() {  
 File file = new File("C:\\Users\\Furkan\\Desktop\\Programming\\JAVA\\workingWithFiles\\src\\testyapmakicin.txt");  
 try {  
 Scanner reader = new Scanner(file);  
 while (reader.hasNextLine()) {  
 String line = reader.nextLine();  
 System.out.println(line);  
 }  
 reader.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Dosya yazma.  
 public static void writeFile() {  
 try {  
 // append true olmazsa dosyaya baştan yazar.  
 BufferedWriter bufferedWriter=new BufferedWriter(new FileWriter("C:\\Users\\Furkan\\Desktop\\Programming\\JAVA\\workingWithFiles\\src\\testyapmakicin.txt",true));  
  
 // Yeni satıra geçer.  
 bufferedWriter.newLine();  
  
 // Dosyaya yazı yazar.  
 bufferedWriter.write("Dosyay");  
  
 System.out.println("Dosyaya yazıldı.");  
  
 // Bağlantıyı kapatır.  
 bufferedWriter.close();  
   
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
}

GENERICS

Main.class

import java.util.ArrayList;  
  
public class Main {  
  
 public static void main(String[] args) {  
 MyList<Customer> sehirler=new MyList<Customer>();  
 sehirler.add(new Customer());  
 sehirler.add(new Customer());  
 }  
}

Customer.class

public class Customer {  
 int id;  
 String firstName;  
}

MyList.class

import java.util.ArrayList;  
  
public class Main {  
  
 public static void main(String[] args) {  
 MyList<Customer> sehirler=new MyList<Customer>();  
 sehirler.add(new Customer());  
 sehirler.add(new Customer());  
 }  
}

THREADING

public class Main {  
  
 public static void main(String[] args) {  
 KronometreThread thread1 = new KronometreThread("thread1");  
  
 KronometreThread thread2 = new KronometreThread("thread2");  
  
 KronometreThread thread3 = new KronometreThread("thread3");  
  
 // Run komutu tek çekirdekle işlem yaparken, start fonksiyonumuz birden çok çekirdekle işlem yapıyor.  
 // Dolayısıyla işlemler neredeyse aynı anda gerçekleşiyor.  
 // Start fonksiyonu da run'ı çalıştırıyor.  
 thread1.start();  
 thread2.start();  
 thread3.start();  
 }  
}

public class KronometreThread implements Runnable{  
  
 private Thread thread;  
 private String threadName;  
  
 public KronometreThread(String threadName){  
 this.threadName = threadName;  
 System.out.println("Oluşturuluyor : "+threadName);  
 }  
  
 @Override  
 public void run() {  
 System.out.println("Çalıştırılıyor : "+threadName);  
  
 try  
 {  
 for (int i=1;i<=10;i++){  
 System.out.println(threadName + " : "+i);  
 Thread.sleep(1000);  
 }  
 }catch (InterruptedException exception){  
 System.out.println("Kesildi : "+threadName);  
 }  
  
 System.out.println("Thread bitti : "+ threadName);  
  
 }  
  
 public void start(){  
 System.out.println("Thread nesnesi oluşuyor");  
 if(thread==null){  
 thread =new Thread(this,threadName);  
 thread.start();  
 }  
 }  
  
}

JDBC

Database Connection

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class Main {  
  
 // Database bilgileri  
 static String userName="root";  
 static String password="1234";  
 static String dbUrl="jdbc:mysql://localhost:3306/world";  
  
 public static void main(String[] args) throws SQLException {  
 Connection connection=null;  
 try  
 {  
 // Database bağlantısı DriveManager ile kuruluyor.  
 connection= DriverManager.getConnection(dbUrl,userName,password);  
  
 System.out.println("Bağlantı oluştu");  
 }  
 catch (SQLException exception)  
 {  
 System.out.println(exception.getMessage());  
 }  
 finally  
 {  
 connection.close();  
 }  
  
 }  
}

DBHELPER AND SQL STATEMENTS

Main.class

import java.sql.\*;  
  
public class Main {  
  
  
 public static void main(String[] args) throws SQLException  
 {  
 // Bağlantı için nesne oluşturduk.  
 Connection connection=null;  
  
 // Sql komutu oluşturmak için nesne oluşturduk.  
 Statement statement=null;  
  
 // Sonuçları almak için nesne oluşturduk.  
 ResultSet resultSet;  
  
 // Database helper'ımızı main'de kullandık.  
 DBHelper helper = new DBHelper();  
  
 try  
 {  
 // connection ve statement nesnelerini doldurduk  
 connection=helper.getConnection();  
 statement=connection.createStatement();  
  
 // Statement'a sorgu gönderdik ve sonuçları resultSet'e atadık.  
 resultSet=statement.executeQuery("select \* from country where name='turkey'");  
  
 // Sonucu continent'in türü string olduğu için getString ile aldık.  
 while(resultSet.next()){  
 System.out.println(resultSet.getString("continent"));  
 }  
  
 }  
  
 catch (SQLException exception)  
 {  
 helper.showErrorMessage(exception);  
 }  
  
 finally  
 {  
 connection.close();  
 }  
  
 }  
}

DBHelper.class

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBHelper {  
 // Database bilgileri  
 private String userName="root";  
 private String password="1234";  
 private String dbUrl="jdbc:mysql://localhost:3306/world";  
  
 // Connection fonksiyonu yazdık  
 public Connection getConnection() throws SQLException {  
 return DriverManager.getConnection(dbUrl,userName,password);  
 }  
  
 public void showErrorMessage(SQLException exception){  
 System.out.println("Error: "+exception.getMessage());  
 System.out.println("Error code: "+exception.getErrorCode());  
 }  
  
 public void closeConnection(){  
  
 }  
}

SELECT, INSERT, UPDATE, DELETE

Main.class

import java.sql.\*;  
import java.util.ArrayList;  
  
public class Main {  
  
  
 public static void main(String[] args) throws SQLException  
 {  
 //selectFromSql();  
  
 //insertData();  
  
 //updateData();  
  
 //deleteData();  
  
 }  
  
 public static void deleteData() throws SQLException {  
 Connection connection=null;  
 PreparedStatement statement=null;  
 ResultSet resultSet;  
 DBHelper helper = new DBHelper();  
  
 try  
 {  
 connection=helper.getConnection();  
  
 String sql="delete from city where id=?";  
  
 statement=connection.prepareStatement(sql);  
 statement.setInt(1,4086);  
 int result =statement.executeUpdate();  
 System.out.println(result+ " tane kayıt silindi");  
 }  
  
 catch (SQLException exception)  
 {  
 helper.showErrorMessage(exception);  
 }  
  
 finally  
 {  
 statement.close();  
 connection.close();  
 }  
 }  
  
 public static void updateData() throws SQLException {  
 Connection connection=null;  
 PreparedStatement statement=null;  
 ResultSet resultSet;  
 DBHelper helper = new DBHelper();  
  
 try  
 {  
 connection=helper.getConnection();  
  
 String sql="update city set name='newKastamonu', district='Karadeniz' where id=?";  
  
 statement=connection.prepareStatement(sql);  
 statement.setInt(1,4086);  
 int result =statement.executeUpdate();  
 System.out.println(result+ " tane kayıt güncellendi");  
 }  
  
 catch (SQLException exception)  
 {  
 helper.showErrorMessage(exception);  
 }  
  
 finally  
 {  
 statement.close();  
 connection.close();  
 }  
 }  
  
 public static void insertData() throws SQLException {  
 Connection connection=null;  
 PreparedStatement statement=null;  
 ResultSet resultSet;  
 DBHelper helper = new DBHelper();  
  
 try  
 {  
 connection=helper.getConnection();  
  
 // prepareStatement createStatement'tan daha performanslıdır.  
 // prepareStatement database'e bir kere sorgu gönderir ve onun üzerinde işlem yaparken,  
 // createStatement her seferinde database'e yeni bir sorgu gönderir.  
  
 // Değerleri dışarıdan atayabiliriz.  
 String sql="insert into city (Name, CountryCode, District,Population) values (?,?,?,?) ";  
  
 statement=connection.prepareStatement(sql);  
 statement.setString(1,"Kastamonu");  
 statement.setString(2,"TUR");  
 statement.setString(3,"Karadeniz");  
 statement.setInt(4,100123);  
 // Kayıt ekler ve kaç kayıt eklediğini döndürür.  
 int result =statement.executeUpdate();  
 System.out.println(result+ " tane kayıt eklendi");  
 }  
  
 catch (SQLException exception)  
 {  
 helper.showErrorMessage(exception);  
 }  
  
 finally  
 {  
 statement.close();  
 connection.close();  
 }  
 }  
  
 public static void selectFromSql() throws SQLException {  
 // Bağlantı için nesne oluşturduk.  
 Connection connection=null;  
  
 // Sql komutu oluşturmak için nesne oluşturduk.  
 Statement statement=null;  
  
 // Sonuçları almak için nesne oluşturduk.  
 ResultSet resultSet;  
  
 // Database helper'ımızı main'de kullandık.  
 DBHelper helper = new DBHelper();  
  
 try  
 {  
 // connection ve statement nesnelerini doldurduk  
 connection=helper.getConnection();  
 statement=connection.createStatement();  
  
 // Statement'a sorgu gönderdik ve sonuçları resultSet'e atadık.  
 resultSet=statement.executeQuery("select \* from country ");  
  
 // Sonuçları içerisine atabilmek için bir country listesi oluşturduk.  
 ArrayList<Country> countries= new ArrayList<Country>();  
  
 // Sonucu continent'in türü string olduğu için getString ile aldık.  
 while(resultSet.next()){  
 countries.add(new Country(  
 resultSet.getString("Code"),  
 resultSet.getString("Name"),  
 resultSet.getString("Continent"),  
 resultSet.getString("Region")  
 ));  
 }  
 System.out.println(countries.size()+ " tane ülke aktarıldı.");  
  
 }  
  
 catch (SQLException exception)  
 {  
 helper.showErrorMessage(exception);  
 }  
  
 finally  
 {  
 statement.close();  
 connection.close();  
 }  
 }  
}

Country.class

public class Country  
{  
 private String code;  
 private String name;  
 private String continent;  
 private String region;  
  
 public Country(String code, String name, String continent, String region)  
 {  
 this.code=code;  
 this.name=name;  
 this.continent=continent;  
 this.region=region;  
 }  
}

DBHelper.class

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DBHelper {  
 // Database bilgileri  
 private String userName="root";  
 private String password="1234";  
 private String dbUrl="jdbc:mysql://localhost:3306/world";  
  
 // Connection fonksiyonu yazdık  
 public Connection getConnection() throws SQLException {  
 return DriverManager.getConnection(dbUrl,userName,password);  
 }  
  
 public void showErrorMessage(SQLException exception){  
 System.out.println("Error: "+exception.getMessage());  
 System.out.println("Error code: "+exception.getErrorCode());  
 }  
  
 public void closeConnection(){  
  
 }  
}

SWING

// Textboxlar

// Textbox textini txb.setText(“Furkan”); ile set edebiliriz.

// Textbox textini txb.getText ile alabiliriz.

// Listeler

// DefaultListModel model= new DefaultListModel(); ile model oluştururuz.

// Daha sonra bu modeli lstOgrenci.setModel(model) ile listeye atarız.

// Bundaki sebep elimizdeki veri setini başka yerlerde de kullanabilmektir.

// model.addElement(“Furkan”) ile veri ekleyebiliriz.

// model.add(5,”Furkan”) ile de veri ekleyebiliriz. Bu komuttaki 5 verinin yer alacağı index numarasını gösterir.

// model.removeElement(“Furkan”) ile veri silinebilir.

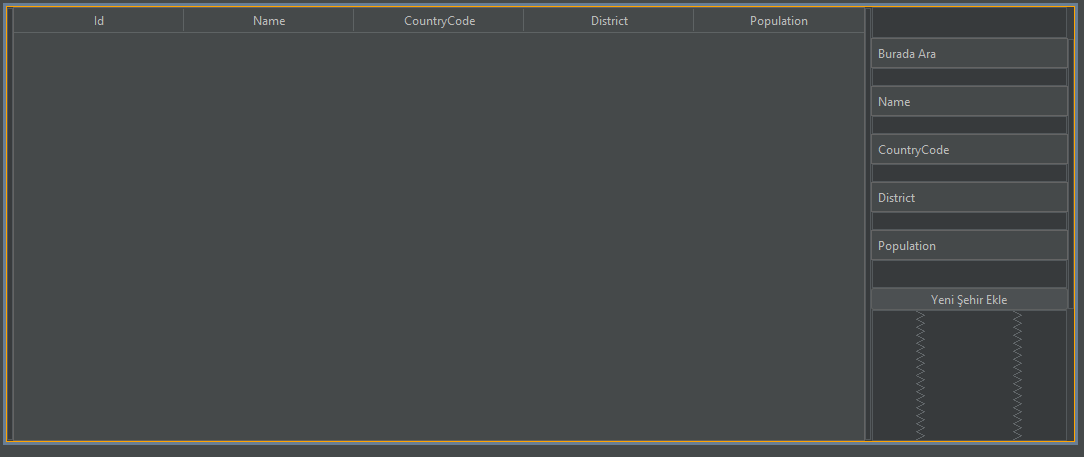
// lstOgrenci.getSelectedIndex() ile listedeki seçili elemanın indexini alabiliriz ve bunu bir değişkene atayabiliriz. Seçilecek eleman yoksa bu komut -1 döndürür.

// Daha sonra bu elemanı model.removeElementAt(değişkenAdı)ile silebiliriz.

// Tablolar

// Sağ tık 🡪 Table Contents 🡪 Columns kısmından database’inize göre title eklenebilir. Title isimleri değiştirilebilir. Sütunlardaki title’ların tipleri girilebilir.

Form.java



public class frm extends javax.swing.JFrame {

// Veriyi içine atabilmek için model oluşturduk.

DefaultTableModel model;

public frm() {

initComponents();

populateTable();

}

public void populateTable(){

model = (DefaultTableModel)tblCities.getModel();

// model'i temizler

model.setRowCount(0);

try {

// cities listesine şehirleri çekip, listeye dönüştürerek döndüren fonksiyonu çağırdık.

ArrayList<City> cities= getCities();

// cities listesindeki her bir satırı model'e ekledik.

for(City city:cities){

Object[] row={

city.getId(),

city.getName(),

city.getCountryCode(),

city.getDistrict(),

city.getPopulation()

};

model.addRow(row);

}

} catch (SQLException ex) {

System.out.println("Hata");

}

}

// Database'deki şehirleri almak için fonksiyon yazdık.

public ArrayList<City> getCities() throws SQLException{

Connection connection=null;

DbHelper dbHelper= new DbHelper();

Statement statement=null;

ResultSet result;

ArrayList<City> cities=null;

try{

connection = dbHelper.getConnection();

statement=connection.createStatement();

result=statement.executeQuery("select \* from city");

// Burada newlememizin sebebi eğer connection çalışmazsa boşuna newlemeyelim diye.

cities = new ArrayList<City>();

// sonuçlar ilerledikçe cities listesine yeni şehir ekle.

while(result.next()){

cities.add(new City(

result.getInt("ID"),

result.getString("Name"),

result.getString("CountryCode"),

result.getString("District"),

result.getInt("Population")

));

}

}

catch(SQLException exception){

dbHelper.showErrorMessage(exception);

}

finally{

statement.close();

connection.close();

}

return cities;

}

private void tbxSearchKeyReleased(java.awt.event.KeyEvent evt) {

String searchKey=tbxSearch.getText();

// TableRowSorter tablo sıralama yapar.

TableRowSorter<DefaultTableModel> tableRowSorter=new TableRowSorter<DefaultTableModel>(model);

// tblCities'e rowsorter olarak tableRowSorter'ı gönderdik.

tblCities.setRowSorter(tableRowSorter);

// tableRowSorter'a filtre olarak searchkey'i gönderdik.

tableRowSorter.setRowFilter(RowFilter.regexFilter(searchKey));

}

private void btnEkleActionPerformed(java.awt.event.ActionEvent evt) {

Connection connection=null;

DbHelper dbHelper= new DbHelper();

PreparedStatement statement=null;

try{

connection=dbHelper.getConnection();

String sql="insert into city(Name, CountryCode, District, Population)values(?,?,?,?)";

statement = connection.prepareStatement(sql);

statement.setString(1,tbxName.getText());

statement.setString(2,tbxCountryCode.getText());

statement.setString(3,tbxDistrict.getText());

statement.setInt(4,Integer.valueOf(tbxPopulation.getText()));

statement.executeUpdate();

populateTable();

}

catch(SQLException exception){

dbHelper.showErrorMessage(exception);

}

finally{

try {

statement.close();

} catch (SQLException ex) {

Logger.getLogger(frm.class.getName()).log(Level.SEVERE, null, ex);

}

try {

connection.close();

} catch (SQLException ex) {

}

DbHelper.java

import java.sql.\*;

public class DbHelper {

private String userName="root";

private String password="1234";

private String dbUrl="jdbc:mysql://localhost:3306/world";

public Connection getConnection() throws SQLException{

return DriverManager.getConnection(dbUrl,userName, password);

}

public void showErrorMessage(SQLException exception){

System.out.println("Error: "+exception.getMessage());

System.out.println("Error code: "+exception.getErrorCode());

}

}

City.java

public class City {

private int id;

private String name;

private String CountryCode;

private String District;

private Integer Population;

public City(int id, String name, String CountryCode, String District, Integer Population) {

this.id = id;

this.name = name;

this.CountryCode = CountryCode;

this.District = District;

this.Population = Population;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCountryCode() {

return CountryCode;

}

public void setCountryCode(String CountryCode) {

this.CountryCode = CountryCode;

}

public String getDistrict() {

return District;

}

public void setDistrict(String District) {

this.District = District;

}

public Integer getPopulation() {

return Population;

}

public void setPopulation(Integer Population) {

this.Population = Population;

}

}

SPRING

Spring ioc with applicationContext.xml

Values.properties

msSqlConnectionString=MsSql ile baglanti kuruldu

mySqlConnectionString=MySql ile baglanti kuruldu

oracleConnectionString=Oracle ile baglanti kuruldu

ApplicationContext.xml

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<!-- values.properties dosyasını applicationContext'e ekledik. -->

<**context:property-placeholder** location=*"/springIntro/values.properties"* />

<!-- ismini dataSource yaptık. class içerisindeki class ı newler. -->

<**bean** id=*"dataSource"* class=*"springIntro.MsSqlCustomerDal"*>

<!-- adı connectionString olan property'ye value içerisindeki değeri atar. -->

<!-- value içerisine values.propertiesdeki msSqlConnectionString'i atadık. ${} kullanarak. -->

<**property** name=*"connectionString"* value=*"${msSqlConnectionString}"*></**property**>

</**bean**>

<**bean** id=*"service"* class=*"springIntro.CustomerManager"*>

<!-- constructor boşsa alttaki satıra gerek yok. hata verir zaten. -->

<**constructor-arg** ref=*"dataSource"*/>

</**bean**>

</**beans**>

IcustomerService.class

package springIntro;

public interface ICustomerService {

void add();

}

CustomerManager.class

package springIntro;

public class CustomerManager implements ICustomerService {

private ICustomerDal customerDal;

public CustomerManager(ICustomerDal customerDal) {

this.customerDal=customerDal;

}

public void add()

{

customerDal.add();

}

}

IcustomerDal.class

package springIntro;

public interface ICustomerDal {

void add();

}

MySqlCustomerDal.class

package springIntro;

public class MySqlCustomerDal implements ICustomerDal {

String connectionString;

public String getConnectionString() {

return connectionString;

}

public void setConnectionString(String connectionString) {

this.connectionString = connectionString;

}

public void add()

{

System.***out***.println(this.connectionString);

System.***out***.println("MySql veritabanına eklendi");

}

}

MsSqlCustomerDal.class

package springIntro;

public class MsSqlCustomerDal implements ICustomerDal {

String connectionString;

public String getConnectionString() {

return connectionString;

}

public void setConnectionString(String connectionString) {

this.connectionString = connectionString;

}

*@Override*

public void add() {

System.***out***.println(this.connectionString);

System.***out***.println("MsSql veritabanına kaydedildi.");

}

}

(AslındaOracle)CustomerDal.class

package springIntro;

public class CustomerDal implements ICustomerDal{

// applicationContext'ten connectionstring set edeceğiz.

String connectionString;

public String getConnectionString() {

return connectionString;

}

public void setConnectionString(String connectionString) {

this.connectionString = connectionString;

}

public void add()

{

System.***out***.println(this.connectionString);

System.***out***.println("Oracle veritabanına eklendi");

}

}

Main.class

package springIntro;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

// applicationContext'i contexte attık.

// ApplicationContext context=new ClassPathXmlApllicationContext şeklinde de tanım yapılabilirdi.

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("/springIntro/applicationContext.xml");

//CustomerManager customerManager= new CustomerManager(new MySqlCustomerDal());

// context'e gidip id ve türe göre newleme yapacak kod satırını çağırır.

//CustomerManager customerManager= new CustomerManager(context.getBean("dataSource",ICustomerDal.class));

//customerManager.add();

// contexten service id li beani buluyor. içerisinde tanımlanan ıcustomerservice türündeki classı newliyor.

// bu bean içerisinde referans olarak dataSource var. Çünkü customermanager constructorı bir customerdal istiyor.

// dataSource bean'i içerisindeki customerDal'ı newleyip parametre olarak customermanagere gönderiyor.

ICustomerService customerService=context.getBean("service",ICustomerService.class);

customerService.add();

}

}

Values.properties

msSqlConnectionString=MsSql ile baglanti kuruldu

mySqlConnectionString=MySql ile baglanti kuruldu

oracleConnectionString=Oracle ile baglanti kuruldu

IoCConfig.config

package com.springdemo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

// dosyanın bir config dosyası olduğunu bildirir.

@Configuration

// com.springdemo'daki tüm componentleri arar.

@ComponentScan("com.springdemo")

// Dosyaları tarayarak @value var mı diye kontrol eder.

// Varsa value içerisindeki değeri values.properties'den bulur ve property'ye atar.

@PropertySource("/com/springdemo/values.properties")

public class IoCConfig {

// bean olduğunu söylemezsek hata verir.

@Bean

public ICustomerDal dataSource() {

return new MsSqlCustomerDal();

}

@Bean

public ICustomerService service() {

return new CustomerManager(dataSource());

}

}

IcustomerService.class

package com.springdemo;

public interface ICustomerService {

void add();

}

CustomerManager.class

package com.springdemo;

public class CustomerManager implements ICustomerService {

private ICustomerDal customerDal;

public CustomerManager(ICustomerDal customerDal) {

this.customerDal=customerDal;

}

public void add()

{

customerDal.add();

}

}

IcustomerDal.class

package com.springdemo;

public interface ICustomerDal {

void add();

}

MySqlCustomerDal.class

package com.springdemo;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

public class MySqlCustomerDal implements ICustomerDal {

@Value("${mySqlConnectionString}")

String connectionString;

public String getConnectionString() {

return connectionString;

}

public void setConnectionString(String connectionString) {

this.connectionString = connectionString;

}

public void add()

{

System.out.println(this.connectionString);

System.out.println("MySql veritabanına eklendi");

}

}

MsSqlCustomerDal.class

package com.springdemo;

import org.springframework.beans.factory.annotation.Value;

public class MsSqlCustomerDal implements ICustomerDal {

// property value'sinin hangi değişkende olduğunu yazdık.

// IoC config'teki @PropertySource'dan çeker.

*@Value*("${msSqlConnectionString}")

String connectionString;

public String getConnectionString() {

return connectionString;

}

public void setConnectionString(String connectionString) {

this.connectionString = connectionString;

}

*@Override*

public void add() {

System.***out***.println(this.connectionString);

System.***out***.println("MsSql veritabanına kaydedildi.");

}

}

Main.class

package com.springdemo;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

//ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("/com/springdemo/applicationContext.xml");

AnnotationConfigApplicationContext context=new AnnotationConfigApplicationContext(IoCConfig.class);

// has not been refreshed yet hatası alınırsa

//context.refresh();

ICustomerService customerService=context.getBean("service",ICustomerService.class);

customerService.add();

}

}

HIBERNATE