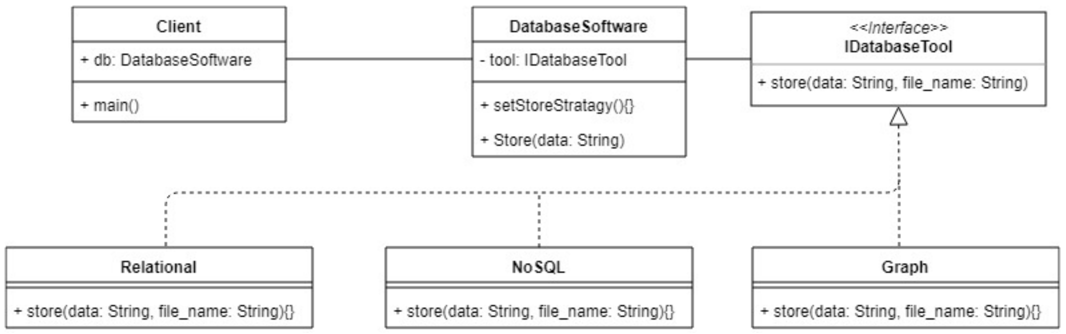
ESOF 322 - Homework 2

Nathan Stouffer and Kevin Browder

September 19, 2019

Part A

Pictured below is our class diagram.


Client
+ db: DatabaseSoftware
+ main()
DatabaseSoftware
- tool: IDatabaseTool
+ setStoreStratagy(){}
+ Store(data: String)
<<Interface>>
IDatabaseTool
+ store(data: String, file_name: String)
Relational
+ store(data: String, file_name: String){}
NoSQL
+ store(data: String, file_name: String){}
Graph
+ store(data: String, file_name: String){}

Part B

Code

```
/**
* interface with a method called store
*/
public interface IDatabaseTool {

    public void store(String data, String file_name);

}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**
* class that implements a dummy database software
*
* A client can switch between different storage tools
* using the setStorageStrategy method
*
* The data will be written to the same file, regardless
* of what storage tool is chosen
*
*/
public class DatabaseSoftware {

    // instance of database tool interface
    private IDatabaseTool tool;
    // output file name
    String file_name = "database-output.out";

    /**
    * constructor for DatabaseSoftware object
    * default database tool is set to relational
    */
    DatabaseSoftware(){
        // create empty file
        createEmptyFile(file_name);

        // dummy output line
        System.out.println("Created Database Software object\nDefault tool is Relational");

        // instantiate tool with Relational constructor
        this.tool = new Relational();
    }

    /**
    * method to store the data using the current store strategy
    * @param data
    */
    public void Store(String data){ tool.store(data, this.file_name); }

    /**
    * method to set a new store strategy
    * @param temp
    */
    public void setStoreStrategy(IDatabaseTool temp){ this.tool = temp; }

    /**
    * method to create an empty output file
    * @param file_name
    */
    private void createEmptyFile(String file_name){
        File file = new File(file_name);
        try{
            FileWriter fw = new FileWriter(file);
            fw.write("");
            fw.close();
        }
        catch (IOException e){ System.err.println("IO error"); }
    }

}
```

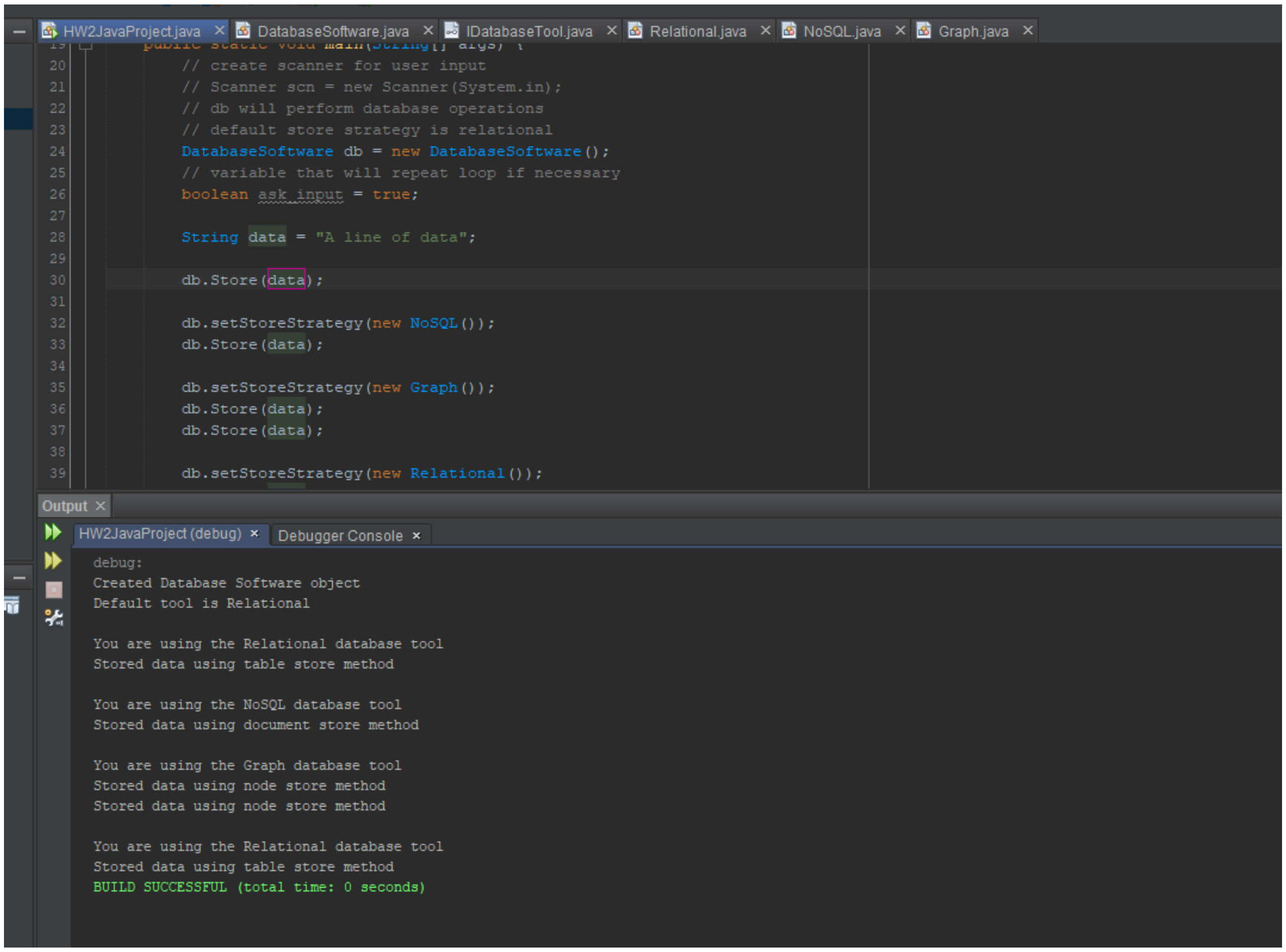
```java
import java.io.FileWriter;
import java.io.File;
import java.io.IOException;

/**
* class that stores data using the Relational tool
* (via that table store method)
*
* class Relational implements the IDatabaseTool interface
*/
public class Relational implements IDatabaseTool {

    /**
    * constructor with a dummy output line to tell user
    * that they are now using the relational tool
    */
    Relational(){ System.out.println("\nYou are using the Relational database tool"); }

    /**
    * method to store the data using the table store method
    * @param data
    * @param file_name
    */
    public void store(String data, String file_name){
        fileOutput(data, file_name);
        System.out.println("Stored data using table store method");
    }

    /**
    * method to write the data to an output file
    * @param data
    * @param file_name
    */
    private void fileOutput(String data, String file_name){
        File file = new File(file_name);
        try{
            FileWriter fw = new FileWriter(file, true);
            fw.write(data);
            fw.close();
        }
        catch (IOException e){ System.err.println("IO error"); }
    }

}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**
* class that stores data using the NoSQL tool
* (via that document store method)
*
* class Relational implements the IDatabaseTool interface
*/
public class NoSQL implements IDatabaseTool {

    /**
    * constructor to output dummy line to tell user
    * that they are now using the NoSQL tool
    */
    NoSQL(){ System.out.println("\nYou are using the NoSQL database tool"); }

    /**
    * method to store data using document store method
    * @param data
    * @param file_name
    */
    public void store(String data, String file_name){
        fileOutput(data, file_name);
        System.out.println("Stored data using document store method");
    }

    /**
    * method to write the data to an output file
    * @param data
    * @param file_name
    */
    private void fileOutput(String data, String file_name){
        File file = new File(file_name);
        try{
            FileWriter fw = new FileWriter(file, true);
            fw.write(data);
            fw.close();
        }
        catch (IOException e){ System.err.println("IO error"); }
    }
}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**
* class that stores data using the Graph tool
* (via that node store method)
*
* class Relational implements the IDatabaseTool interface
*/
public class Graph implements IDatabaseTool {

    /**
    * constructor to output dummy line to tell user
    * that they are now using the NoSQL tool
    */
    Graph(){
         System.out.println("\nYou are using the Graph database tool");
    }

    /**
    * method to store data using the node store method
    * @param data
    * @param file_name
    */
    public void store(String data, String file_name){
        fileOutput(data, file_name);
        System.out.println("Stored data using node store method");
    }

    /**
    * method to write the data to an output file
    * @param data
    * @param file_name
    */
    private void fileOutput(String data, String file_name){
        File file = new File(file_name);
        try{
            FileWriter fw = new FileWriter(file, true);
            fw.write(data);
            fw.close();
        }
        catch (IOException e){ System.err.println("IO error"); }
    }
}
```

Output

Our console output is shown below. The console output consists of print statements from methods. In addition to these print statements, the string ”A line of data” is written to a file whenever the *Store()* method is called on a DatabaseSoftware object.

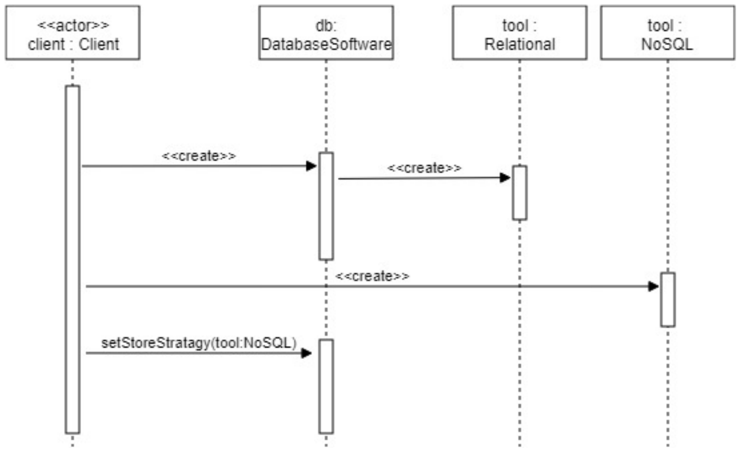


```
    public static void main(String[] args) {
        // create scanner for user input
        // Scanner scn = new Scanner(System.in);
        // db will perform database operations
        // default store strategy is relational
        DatabaseSoftware db = new DatabaseSoftware();
        // variable that will repeat loop if necessary
        boolean ask_input = true;

        String data = "A line of data";

        db.Store(data);

        db.setStoreStrategy(new NoSQL());
        db.Store(data);

        db.setStoreStrategy(new Graph());
        db.Store(data);
        db.Store(data);

        db.setStoreStrategy(new Relational());
```

```
debug:
Created Database Software object
Default tool is Relational

You are using the Relational database tool
Stored data using table store method

You are using the NoSQL database tool
Stored data using document store method

You are using the Graph database tool
Stored data using node store method
Stored data using node store method

You are using the Relational database tool
Stored data using table store method
BUILD SUCCESSFUL (total time: 0 seconds)
```

Part C

Pictured below is our sequence diagram. The sequence diagram pictures how our code switches between two strategies.


<<actor>>
client : Client
db:
DatabaseSoftware
tool :
Relational
tool :
NoSQL
<<create>>
<<create>>
<<create>>
setStoreStratagy(tool:NoSQL)