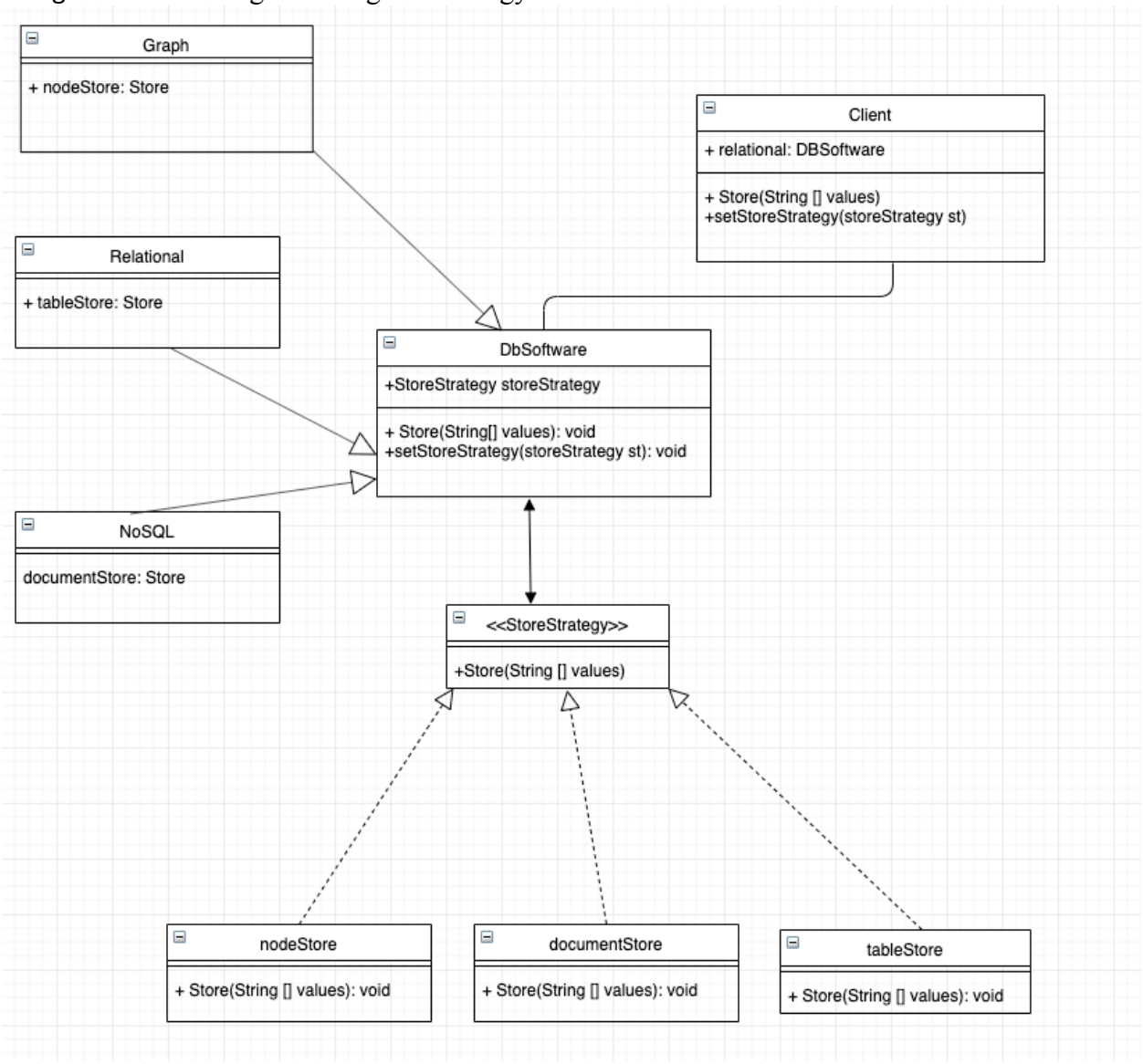
Homework2:
Abdulrahman Alhitmi & Ahmed Naji

A) Design UML class diagram using the Strategy Pattern:



B) Write code to implement the design:

Client Class:

```
import java.util.Arrays;
//this is the client class (main class)
public class Client{
        public static void main(String[] args){
                //initialize the first and default method.
                DbSoftware relational = new Relational();
                //the array list we got
```

```
                String names[] = new String []{"Arron","Jacob","Kate","Mike"};
                System.out.println("This is the list of names: ");
                System.out.println(names);

                //the first method is relational db
                System.out.println("Table Store: ");
                relational.Store(names);
                //second method switching to is NoSQL
                relational.setStoreStrategy(new documentStore());
                System.out.println("document Store: ");
                names = new String []{"Arron","Jacob","Kate","Mike"};
                relational.Store(names);

                //last method is Graph method which was weird to create especially with
names
                relational.setStoreStrategy(new nodeStore());
                System.out.println("node Store");
                names = new String []{"Arron","Jacob","Kate","Mike"};
                relational.Store(names);
        }
}
```

NoSQL Class:

```
public class NoSQL extends DbSoftware{
        public NoSQL(){
                storeStrategy = new documentStore();
        }
}
```

Graph Class:

```
public class Graph extends DbSoftware{
        public Graph(){
                storeStrategy = new nodeStore();
        }
}
```

Relational Class:

```
public class Relational extends DbSoftware{
        public Relational(){
                storeStrategy = new tableStore();
        }
}
```

Document Store Class:

```
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class documentStore implements StoreStrategy{
	public void Store(String[] values){
		try{
			PrintWriter out = new PrintWriter(new BufferedWriter(new
FileWriter("doc2.txt")));

			for(int x = 0; x < values.length; x++){

				out.write(values[x]);
				out.write("\n");
				System.out.println(values[x]);
			}
			out.close();
		}
		catch(IOException e){
			System.out.println(e);
		}
	}
}
```

Table Store Class:

```
public class tableStore implements StoreStrategy{
	public void Store(String[] values){
		System.out.println("Index\tValue");
		for(int x = 0; x < values.length; x++){
			System.out.println(x + "\t" + values[x]);
		}
	}
}
```

DbSoftware Class:

```
public class DbSoftware{
	StoreStrategy storeStrategy;

	public DbSoftware(){
	}

	public void Store(String[] values){
		storeStrategy.Store(values);
```

```
        }

        public void setStoreStrategy(StoreStrategy st){
                storeStrategy = st;
        }
}
```

Store Strategy Class:

```
public interface StoreStrategy{
        public void Store(String[] values);
}
```

Main Output:

C) Create UML sequence diagram to show creation/usage of any tool, the sorting execution, and runtime of switching algorithms: