Part B: Code

```
import java.util.Scanner;

public class Main {
   static Scanner myScanner = new Scanner(System.in);

   public static void main(String[] args){

   // Get user input to select which type of defualt database to create

      String clientChoice="0";

      do {
         System.out.print("Choose a Database type by entering the cooresponding integer:\n" +
               "[1]    Relational\n" +
               "[2]    NoSQL\n" +
               "[3]    Graph\n" +
               "=> ");
         try {
            clientChoice = myScanner.next();
         } catch (Exception e){
            System.out.println("scanner error");
         }

      }while (!(clientChoice.equals("1") || clientChoice.equals("2") || clientChoice.equals("3")));

      System.out.println("");


   // Create the DB based on client selection

      DatabaseSoftware myDB= null;

      switch (clientChoice){
         case "1":
            System.out.println("Relational Database Selected");
            myDB = new RelationalDataBase();
            //DBActions(myRelDB);
            //myRelDB.performStore();
            break;

         case "2":
            System.out.println("NoSQL Database Selected");
            myDB = new NoSQLDataBase();

            //DBActions(myNoSQL);
            //myNoSQL.performStore();

            break;

         case "3":
            System.out.println("Graph Database Selected");
            myDB = new GraphDatabase();
            //DBActions(myGraph);

            break;
      }

   // Get user input for next actions
   //    eg. enter data or change storage strategy
```

```
        String DBAction = "0";
        String data;

        do{
            System.out.print("Choose an action by entering an integer:\n" +
                    "[1]    Enter New Data\n" +
                    "[2]    Change Storage Strategy\n" +
                    "[9]    Quit\n" +
                    "=> ");
            DBAction = myScanner.next();

            if (DBAction.equals("1")){
                System.out.println("Enter data: ");
                myScanner.nextLine();
                data = myScanner.nextLine();
                myDB.performStore(data);

            } else if (DBAction.equals("2")) {
                do {
                    System.out.println(("Enter new strategy (NoSQL, Relational, Graph"));
                    data = myScanner.next();
                } while (!(data.equalsIgnoreCase("nosql") ||
                        data.equalsIgnoreCase("relational") || data.equalsIgnoreCase("graph")));

                System.out.println("Changing database strategy to: " + data);

                if (data.equalsIgnoreCase("nosql")) {
                    myDB.setStoreBehavior(new DocumentStore());

                } else if (data.equalsIgnoreCase("relational")) {
                    myDB.setStoreBehavior(new TableStore());

                } else {
                    myDB.setStoreBehavior(new NodeStore());
                }
            } else {
                System.out.println("Exiting...");
            }
        } while (!(DBAction.equals("9")));

    }
}
```

```
public abstract class DatabaseSoftware {

    // fields:
    StoreBehavior storeBehavior;

    // constructor
    public  DatabaseSoftware(){
    }

    // Methods
    public void performStore(String newData){
        storeBehavior.store(newData);
    }

    public void setStoreBehavior(StoreBehavior newBehavior){
        storeBehavior = newBehavior;
```

```
    }
}
```

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

public class DocumentStore implements StoreBehavior {

    @Override
    public void store(String newData) {
        System.out.println("Storing " + newData + " to a Document...");


        // write to a file
        try {
            File outputFile = new File("DocumentStore.txt");
            if (outputFile.createNewFile()) {
                System.out.println("A new Document file has been created");
            }

            FileWriter outFile1 = new FileWriter(outputFile, true);
            outFile1.append(newData + "\n");
            outFile1.flush();
            outFile1.close();

        } catch (Exception e) {
            System.out.println("Exception writing reciept");
            e.printStackTrace();
        }

    }
}
```

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

public class NodeStore implements StoreBehavior{

    @Override
    public void store(String newData) {
        System.out.println("Storing " + newData + " to a Node....");

        // Write to a file
        try {
            File outputFile = new File("NodeStore.txt");
            if (outputFile.createNewFile()) {
                System.out.println("A new Node file has been created");
            }

            FileWriter outFile1 = new FileWriter(outputFile, true);
            outFile1.append(newData + "\n");
            outFile1.flush();
            outFile1.close();

        } catch (Exception e) {
            System.out.println("Exception writing reciept");
            e.printStackTrace();
        }
    }
}
```

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

public class TableStore implements StoreBehavior {

    @Override
    public void store(String newData) {
        System.out.println("Storing " + newData + " to a table...");


        //Write to a file
        try {
            File outputFile = new File("TableStore.txt");
            if (outputFile.createNewFile()) {
                System.out.println("A new Table file has been created");
            }

            FileWriter outFile1 = new FileWriter(outputFile, true);
            outFile1.append(newData + "\n");
            outFile1.flush();
            outFile1.close();

        } catch (Exception e) {
            System.out.println("Exception writing reciept");
            e.printStackTrace();
        }

    }
}
```

```java
public class GraphDatabase extends DatabaseSoftware {

    // inherited Fields:
    //      StoreBehavior storeBehavior;

    // Constructor

    public GraphDatabase(){
        storeBehavior = new NodeStore();
    }
}
```

```java
public class NoSQLDataBase extends DatabaseSoftware {

    // inherited Fields:
    //      StoreBehavior storeBehavior;

    // Constructor: when this class gets instatiated, the constructor will set
    //  storeBehavior to DocumentStore

    public NoSQLDataBase(){
        storeBehavior = new DocumentStore();
    }

}
```

```
public class RelationalDataBase extends DatabaseSoftware {

   // inherited Fields:
   //      StoreBehavior storeBehavior;

   // constructor
   public RelationalDataBase(){
      storeBehavior = new TableStore();
   }
}
```

```
public interface StoreBehavior {

   public void store(String data);

}
```