```java
import java.util.Scanner;

class ESOF322_HW2_PartB {
    // "Client" implementation.
    public static void main(String[] args) {
        // StdIn scanner. Used by menuing system only.
        Scanner scan = new Scanner(System.in);

        System.out.println("ESOF322, Homework 2, Tyler Ross & Daniel Vinogradov\n");

        // Database instance with Integer typed keys and String typed values.
        // initialSetup() prompts user to select a storage strategy,
        //  or accept the default strategy, and returns a constructed Database.
        Database<Integer,String> db = initialSetup(scan);

        // Exit conditions handled by mainMenu()
        while(true) {
            // mainMenu() provides database "write" functionality, and
            //  runtime changes to storage strategy.
            mainMenu(scan, db);
        }
    }

    // Prompts user to select a storage strategy, or accept the default strategy.
    public static Database<Integer,String> initialSetup(Scanner scan) {
        System.out.println("Please choose from the following options:");
        System.out.println("0. Exit program.");
        System.out.println("1. Construct Database with default storage strategy.");
        System.out.println("2. Construct Database with selected storage strategy.");
        System.out.print("Number of selection: ");

        int selection = -1;
        try {
            String input = scan.nextLine();
            selection = Integer.parseInt(input);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please provide only a number.");
            return initialSetup(scan);
        }

        switch (selection) {
            case 0:
                System.out.println("Exiting.");
                System.exit(0);
            case 1:
                return new Database<>();
            case 2:
                return new Database<>(selectStorageStrategy(scan));
            default:
                System.out.println("Invalid selection! " + selection + " is not an
                option.");
                return initialSetup(scan);
        }
    }

    public static IStorage<Integer,String> selectStorageStrategy(Scanner scan) {
        System.out.println("\nPlease select a Database storage strategy:");
        System.out.println("0. Exit program.");
        System.out.println("1. Relational");
        System.out.println("2. NoSQL");
        System.out.println("3. Graph");
        System.out.print("Number of selection: ");

        int selection = -1;
        try {
            String input = scan.nextLine();
```

```
66              selection = Integer.parseInt(input);
67          } catch (NumberFormatException e) {
68              System.out.println("Invalid input. Please provide only a number.");
69              return selectStorageStrategy(scan);
70          }
71
72          switch (selection) {
73              case 0:
74                  System.out.println("Exiting.");
75                  System.exit(0);
76              case 1:
77                  return new RelationalDB<Integer,String>();
78              case 2:
79                  return new NoSQLDB<Integer,String>();
80              case 3:
81                  return new GraphDB<Integer,String>();
82              default:
83                  System.out.println("Invalid selection! " + selection + " is not an
                    option.");
84                  return selectStorageStrategy(scan);
85          }
86      }
87
88      // Provides options to write to the database and change storage strategies.
89      public static void mainMenu(Scanner scan, Database<Integer,String> db) {
90          System.out.println("\nPlease choose from the following options:");
91          System.out.println("0. Exit program.");
92          System.out.println("1. Write to database");
93          System.out.println("2. Change database storage strategy.");
94          System.out.print("Number of selection: ");
95
96          int selection = -1;
97          try {
98              String input = scan.nextLine();
99              selection = Integer.parseInt(input);
100         } catch (NumberFormatException e) {
101             System.out.println("Invalid input. Please provide only a number.");
102             return;
103         }
104
105         switch (selection) {
106             case 0:
107                 System.out.println("Exiting.");
108                 System.exit(0);
109             case 1:
110                 writeToDB(scan, db);
111                 break;
112             case 2:
113                 db.setStorageStrategy(selectStorageStrategy(scan));
114                 break;
115             default:
116                 System.out.println("Invalid selection! " + selection + " is not an
                    option.");
117         }
118
119     }
120
121     public static void writeToDB(Scanner scan, Database<Integer,String> db) {
122         System.out.print("\nPlease provide an Integer key: ");
123
124         int key;
125         try {
126             String input = scan.nextLine();
127             key = Integer.parseInt(input);
128         } catch (NumberFormatException e) {
129             System.out.println("Invalid input. Key must be Integer.");
```

```
130                 return;
131             }
132
133             System.out.print("Please provide the String to be written: ");
134
135             String value = scan.nextLine();
136
137             db.store(key, value);
138         }
139     }
140
141     /*
142      * Client code ends here.
143      * Everything from here on is strategy-related code.
144      */
145
146     class Database<K,V> {
147         private IStorage<K,V> storageStrategy;
148
149         public Database() {
150             // Initialize with default strategy.
151             this.storageStrategy = new RelationalDB<K,V>();
152             System.out.println("Constructed database instance with default storage strategy
                ("
153                 + this.storageStrategy.getClass().getSimpleName() + ").");
154         }
155
156         public Database(IStorage<K,V> storageStrategy) {
157             // Initialize with given strategy.
158             this.storageStrategy = storageStrategy;
159             System.out.println("Constructed database instance with given storage strategy ("
160                 + this.storageStrategy.getClass().getSimpleName() + ").");
161         }
162
163         public void store(K key, V value) {
164             storageStrategy.store(key, value);
165         }
166
167         public void setStorageStrategy(IStorage<K,V> storageStrategy) {
168             String oldStrat = this.storageStrategy.getClass().getSimpleName();
169             // In reality, there should probably be some sort of data migration here
170             this.storageStrategy = storageStrategy;
171             System.out.println("Updated storage strategy from " + oldStrat
172                 + " to " + this.storageStrategy.getClass().getSimpleName());
173         }
174     }
175
176     interface IStorage<K,V> {
177         public abstract void store(K key, V value);
178     }
179
180     class RelationalDB<K,V> implements IStorage<K,V>{
181
182         @Override
183         public void store(K key, V value) {
184             // Dummy method; prints key, value, and storageStrategy.
185             // Actual storage implementation not requested in assignment spec.
186             System.out.println("Stored " + value.getClass().getSimpleName() + " value '" +
                value
187                 + "' with " + key.getClass().getSimpleName() +" key '" + key
188                 + "' in " + this.getClass().getSimpleName());
189         }
190
191     }
192
193     class NoSQLDB<K,V> implements IStorage<K,V>{
```

```
194
195         @Override
196         public void store(K key, V value) {
197             // Dummy method; prints key, value, and storageStrategy.
198             // Actual storage implementation not requested in assignment spec.
199             System.out.println("Stored " + value.getClass().getSimpleName() + " value '" +
                value
200                 + "' with " + key.getClass().getSimpleName() +" key '" + key
201                 + "' in " + this.getClass().getSimpleName());
202         }
203
204     }
205
206     class GraphDB<K,V> implements IStorage<K,V>{
207
208         @Override
209         public void store(K key, V value) {
210             // Dummy method; prints key, value, and storageStrategy.
211             // Actual storage implementation not requested in assignment spec.
212             System.out.println("Stored " + value.getClass().getSimpleName() + " value '" +
                value
213                 + "' with " + key.getClass().getSimpleName() +" key '" + key
214                 + "' in " + this.getClass().getSimpleName());
215         }
216
217     }
218
```