

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using System.Data.SQLite;
5 using Microsoft.Xna.Framework.Graphics;
6 using System.Threading;
7 using Microsoft.Xna.Framework;
8
9 namespace coursework
10 {
11     class DBManager
12     {
13         public static List<string> commands = new List<string>();
14         /// <summary>
15         /// connect to the database
16         /// </summary>
17         /// <returns></returns>
18         public static SQLiteConnection CreateConnection()
19         {
20
21             SQLiteConnection sqlite_conn;
22             // Create a new database connection:
23             sqlite_conn = new SQLiteConnection("Data Source= Database.db;
24             Version = 3; New = True; Compress = True; ");
25             // Open the connection:
26             try
27             {
28                 sqlite_conn.Open();
29             }
30             catch
31             {
32             }
33             return sqlite_conn;
34         }
35
36         /// <summary>
37         /// if the command queue is not empty, calls RunCommand
38         /// if command queue is empty, thread sleeps until a new command is
39         added
40         /// </summary>
41         public static void commandQueue()
42         {
43             while (Game1.GameActive)
44             {
45                 if (commands.Count != 0)
46                 {
47                     RunCommand(commands[0]);
48                     commands.RemoveAt(0);
49                 }
50                 else
51                 {
52                     try
53                     {
54                         Thread.Sleep(Timeout.Infinite);
55                     }
56                 }
57             }
58         }
59     }
60 }
```

```

55         catch
56         {
57             //sleep interrupted
58         }
59     }
60 }
61 }
62
63 /// <summary>
64 /// when database is unlocked first command in commands list is run
65 /// </summary>
66 /// <param name="cmd"></param>
67 public static void RunCommand(string cmd)
68 {
69     SQLiteConnection conn;
70     SQLiteCommand sqlite_cmd;
71     conn = CreateConnection();
72     sqlite_cmd = conn.CreateCommand();
73     sqlite_cmd.CommandText = cmd;
74     WaitForDbToBeUnlocked();
75     try
76     {
77         sqlite_cmd.ExecuteNonQuery();
78         conn.Close();
79     }
80     catch
81     {
82     }
83 }
84 }
85
86 /// <summary>
87 /// returns true if database is locked
88 /// </summary>
89 /// <returns></returns>
90 public static bool IsDatabaseLocked()
91 {
92     bool locked = true;
93     CreateConnection();
94     SQLiteConnection connection = new SQLiteConnection($"Data Source= Database.db;Version=3;");
95     connection.Open();
96
97     try
98     {
99         SQLiteCommand beginCommand = connection.CreateCommand();
100         beginCommand.CommandText = "BEGIN EXCLUSIVE"; // tries to
101             //
102             // CommandTimeout is set to 0 to get error immediately if DB
103             // is locked
104             // otherwise it
105             // will wait for 30 sec by default
106         beginCommand.CommandTimeout = 0;
107         beginCommand.ExecuteNonQuery();

```

```
106         SQLiteCommand commitCommand = connection.CreateCommand();
107         commitCommand.CommandText = "COMMIT"; // releases the lock
108             immediately
109         commitCommand.ExecuteNonQuery();
110         locked = false;
111     }
112     catch (SQLiteException)
113     {
114         // database is locked error
115     }
116     finally
117     {
118         connection.Close();
119     }
120     return locked;
121 }
122
123 /// <summary>
124 /// calls IsDatabaseLocked every 1 millisecond until it returns false
125 /// </summary>
126 /// <returns></returns>
127 public static bool WaitForDbToBeUnlocked()
128 {
129     while (IsDatabaseLocked())
130     {
131         Thread.Sleep(1);
132     }
133     return true;
134 }
135
136 /// <summary>
137 /// creates database with starting data set
138 /// </summary>
139 /// <param name="conn"></param>
140 public static void InitialiseDB(SQLiteConnection conn)
141 {
142
143     SQLiteCommand sqlite_cmd;
144     string CreateChar = "CREATE TABLE Characters (CharacterID INTEGER,
145         Name VARCHAR(20), Level INT, Race VARCHAR(20), Dexterity
146         INTEGER, Strength INTEGER, ArmorClass INTEGER, UserID INTEGER,
147         PRIMARY KEY(CharacterID AUTOINCREMENT))";
148     string CreateWeapons = "CREATE TABLE Weapons (WeaponName VARCHAR
149         (20), Range INT, hitDie INT)";
150     string CreateMonsters = "CREATE TABLE Monsters (MonsterID INTEGER,
151         MonsterName VARCHAR(20), ArmorClass INTEGER, Strength INTEGER,
152         Dex INTEGER, HitPoints INTEGER, HitDie INTEGER, Range INTEGER,
153         PRIMARY KEY(MonsterID AUTOINCREMENT))";
154     string CreateRace = "CREATE TABLE Race (RaceName VARCHAR(20),
155         DexModifier INTEGER, StrengthModifier INTEGER, Speed INTEGER,
156         PRIMARY KEY(RaceName))";
157     string CreateWeaponAssignment = "CREATE TABLE
158         MonsterWeaponAssignment (MonsterID INTEGER, WeaponID INTEGER,
159         PRIMARY KEY(WeaponID, MonsterID))";
```

```

150         sqlite_cmd = conn.CreateCommand();
151         sqlite_cmd.CommandText = CreateChar;
152         sqlite_cmd.ExecuteNonQuery();
153         sqlite_cmd.CommandText = CreateWeapons;
154         sqlite_cmd.ExecuteNonQuery();
155         sqlite_cmd.CommandText = CreateMonsters;
156         sqlite_cmd.ExecuteNonQuery();
157         sqlite_cmd.CommandText = CreateRace;
158         sqlite_cmd.ExecuteNonQuery();
159         sqlite_cmd.CommandText = CreateWeaponAssignment;
160         sqlite_cmd.ExecuteNonQuery();
161
162         sqlite_cmd.CommandText = "INSERT INTO Weapons (WeaponName, Range,
163             hitDie) VALUES('dagger', 20 , 4);";
164         sqlite_cmd.ExecuteNonQuery();
165         sqlite_cmd.CommandText = "INSERT INTO Weapons (WeaponName, Range,
166             hitDie) VALUES('longbow', 150 , 8);";
167         sqlite_cmd.ExecuteNonQuery();
168
169         sqlite_cmd.CommandText = "INSERT INTO Monsters (MonsterName,
170             ArmorClass, Strength, Dex, HitPoints, HitDie, Range) VALUES
171             ('Awakened Shrub', 9, 3, 8, 10, 4, 9);";
172         sqlite_cmd.ExecuteNonQuery();
173
174         sqlite_cmd.CommandText = "INSERT INTO Monsters (MonsterName,
175             ArmorClass, Strength, Dex, HitPoints) VALUES('Goblin', 15, 8,
176             14, 7);";
177         sqlite_cmd.ExecuteNonQuery();
178
179         sqlite_cmd.CommandText = "INSERT INTO MonsterWeaponAssignment
180             (MonsterID, WeaponID) VALUES( 2, 5);";
181         sqlite_cmd.ExecuteNonQuery();
182         sqlite_cmd.CommandText = "INSERT INTO MonsterWeaponAssignment
183             (MonsterID, WeaponID) VALUES( 2, 4);";
184         sqlite_cmd.ExecuteNonQuery();
185
186         sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier,
187             StrengthModifier, Speed) VALUES('Human', 1, 1, 30);";
188         sqlite_cmd.ExecuteNonQuery();
189         sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier,
190             StrengthModifier, Speed) VALUES('Elf', 2, 0, 30);";
191         sqlite_cmd.ExecuteNonQuery();
192         sqlite_cmd.CommandText = "INSERT INTO Race (RaceName, DexModifier,
193             StrengthModifier, Speed) VALUES('Dwarf', 0, 2, 25);";
194         sqlite_cmd.ExecuteNonQuery();
195     }
196
197     /// <summary>
198     /// creates insert SQL command for a character
199     /// adds command to commands list
200     /// </summary>
201     /// <param name="name"></param>
202     /// <param name="race"></param>
203     /// <param name="level"></param>

```

```
194     /// <param name="dex"></param>
195     /// <param name="str"></param>
196     /// <param name="AC"></param>
197     /// <param name="userID"></param>
198     public static void InsertData(string name, string race, int level,
199     string dex, string str, int AC, string userID)
200     {
201         string cmd = "INSERT INTO Characters (Name, Level, Race,
202         Dexterity, Strength, ArmorClass, UserID) VALUES('" + name + "',
203         " + level + ", '" + race + "', " + dex + ", " + str + ", " + AC
204         + ", '" + userID + "')";
205         commands.Add(cmd);
206     }
207
208     #region read methods
209     /// <summary>
210     /// runs read command and returns resulting data
211     /// </summary>
212     /// <param name="cmd"></param>
213     /// <returns></returns>
214     public static SQLiteDataReader ExecuteReader(string cmd)
215     {
216         SQLiteDataReader data;
217         SQLiteConnection conn;
218         SQLiteCommand sqlite_cmd;
219         conn = CreateConnection();
220         sqlite_cmd = conn.CreateCommand();
221         sqlite_cmd.CommandText = cmd;
222         data = sqlite_cmd.ExecuteReader();
223         return data;
224     }
225
226     /// <summary>
227     /// reads character information from data base where the username
228     matches the users input
229     /// outputs the results
230     /// </summary>
231     /// <param name="conn"></param>
232     /// <param name="spriteBatch"></param>
233     /// <param name="font"></param>
234     /// <param name="username"></param>
235     public static void ReadCharacters(SQLiteConnection conn, SpriteBatch
236     spriteBatch, SpriteFont font, string username)
237     {
238         int count = 0;
239         SQLiteDataReader sqlite_datareader;
240         string cmd = "SELECT * FROM Characters WHERE Characters.UserID =
241         '" + username + "'";
242         int height = 100;
243
244         sqlite_datareader = DBManager.ExecuteReader(cmd);
245
246         spriteBatch.DrawString(font, "ID", new Vector2(50, 50),
247         Color.Pink);
```

```
242         string col2 = sqlite_datareader.GetName(1);
243         spriteBatch.DrawString(font, col2, new Vector2(125, 50),
            Color.Pink);
244
245         string col3 = sqlite_datareader.GetName(2);
246         spriteBatch.DrawString(font, col3, new Vector2(250, 50),
            Color.Pink);
247
248         string col4 = sqlite_datareader.GetName(3);
249         spriteBatch.DrawString(font, col4, new Vector2(350, 50),
            Color.Pink);
250
251         string col5 = sqlite_datareader.GetName(4);
252         spriteBatch.DrawString(font, "Dex", new Vector2(450, 50),
            Color.Pink);
253
254         string col6 = sqlite_datareader.GetName(5);
255         spriteBatch.DrawString(font, "Str", new Vector2(550, 50),
            Color.Pink);
256
257         string col7 = sqlite_datareader.GetName(6);
258         spriteBatch.DrawString(font, "AC", new Vector2(650, 50),
            Color.Pink);
259
260
261         while (sqlite_datareader.Read()) // closed too soon
262         {
263             if (count - (6 * Game1.pageNum) < 6 && count - (6 *
                Game1.pageNum) >= 0)
264             {
265                 string myreader0 = sqlite_datareader.GetInt32(0).ToString
                ();
266                 string myreader1 = sqlite_datareader.GetString(1);
267                 string myreader2 = sqlite_datareader.GetInt32(2).ToString
                ();
268                 string myreader3 = sqlite_datareader.GetString(3);
269                 string myreader4 = sqlite_datareader.GetInt32(4).ToString
                ();
270                 string myreader5 = sqlite_datareader.GetInt32(5).ToString
                ();
271                 string myreader6 = sqlite_datareader.GetInt32(6).ToString
                ();
272
273                 spriteBatch.DrawString(font, myreader0, new Vector2(50,
                height), Color.Pink);
274                 spriteBatch.DrawString(font, myreader1, new Vector2(125,
                height), Color.Pink);
275                 spriteBatch.DrawString(font, myreader2, new Vector2(250,
                height), Color.Pink);
276                 spriteBatch.DrawString(font, myreader3, new Vector2(350,
                height), Color.Pink);
277                 spriteBatch.DrawString(font, myreader4, new Vector2(450,
                height), Color.Pink);
278                 spriteBatch.DrawString(font, myreader5, new Vector2(550,
                height), Color.Pink);
279                 spriteBatch.DrawString(font, myreader6, new Vector2(650,
```

```
        height), Color.Pink);
        height += 50;
    }
    count++;
}

/// <summary>
/// reads all weapon data from the database and outputs it
/// </summary>
/// <param name="conn"></param>
/// <param name="spriteBatch"></param>
/// <param name="font"></param>
public static void ReadWeapons(SQLiteConnection conn, SpriteBatch
    spriteBatch, SpriteFont font)
{
    int count = 0;
    SQLiteDataReader sqlite_datareader;
    SQLiteCommand sqlite_cmd;
    sqlite_cmd = conn.CreateCommand();
    sqlite_cmd.CommandText = "SELECT * FROM Weapons";
    int height = 100;
    sqlite_datareader = sqlite_cmd.ExecuteReader();

    spriteBatch.DrawString(font, "ID", new Vector2(50, 50),
        Color.Pink);

    string col2 = sqlite_datareader.GetName(1);
    spriteBatch.DrawString(font, col2, new Vector2(125, 50),
        Color.Pink);

    string col3 = sqlite_datareader.GetName(2);
    spriteBatch.DrawString(font, col3, new Vector2(250, 50),
        Color.Pink);

    string col4 = sqlite_datareader.GetName(3);
    spriteBatch.DrawString(font, col4, new Vector2(350, 50),
        Color.Pink);

    while (sqlite_datareader.Read())
    {
        if (count - (6 * Game1.pageNum) < 6 && count - (6 *
            Game1.pageNum) >= 0)
        {
            string myreader0 = sqlite_datareader.GetInt32(0).ToString
                ();
            string myreader1 = sqlite_datareader.GetString(1);
            string myreader2 = sqlite_datareader.GetInt32(2).ToString
                ();
            string myreader3 = sqlite_datareader.GetInt32(3).ToString
                ();

            spriteBatch.DrawString(font, myreader0, new Vector2(50,
                height), Color.Pink);
```

```
325         spriteBatch.DrawString(font, myreader1, new Vector2(125, height), Color.Pink);
326         spriteBatch.DrawString(font, myreader2, new Vector2(250, height), Color.Pink);
327         spriteBatch.DrawString(font, myreader3, new Vector2(350, height), Color.Pink);
328         height += 50;
329     }
330     count++;
331 }
332 }
333
334 public static void ReadModifiers(SQLiteConnection conn, ref int dexMod, ref int strMod, string race, ref int speed)
335 {
336     SQLiteDataReader sqlite_datareader;
337     SQLiteCommand sqlite_cmd;
338     sqlite_cmd = conn.CreateCommand();
339     sqlite_cmd.CommandText = "SELECT DexModifier, StrengthModifier, Speed FROM Race WHERE RaceName = '" + race + "'";
340     sqlite_datareader = sqlite_cmd.ExecuteReader();
341     if (sqlite_datareader.Read())
342     {
343         dexMod = sqlite_datareader.GetInt32(0);
344         strMod = sqlite_datareader.GetInt32(1);
345         speed = sqlite_datareader.GetInt32(2);
346     }
347 }
348 #endregion
349 }
350 }
351
```