

```
1  using CustomProject.Command;
2  using CustomProject.Observers;
3  using CustomProject.SingletonDesign;
4  using CustomProject.StrategyDesign;
5  using SplashKitSDK;
6
7  namespace CustomProject
8  {
9      public class GameProcessor
10     {
11         private readonly Window _window;
12         private GameMap _map;
13         private Player _player;
14
15         private Bitmap _interface;
16
17         private string _playerName;
18         private string _playerDescription;
19
20         private bool _gameStarted;
21         private bool _gameOver;
22         private bool _showingInstructions;
23
24         private SwordUnlocked _swordUnlockedNotification;
25         private FastMovementUnlocked _fastMovementNotification;
26         private HighJumpUnlocked _highJumpNotification;
27
28         public GameProcessor()
29         {
30             GetUserName();
31             GetUserDescription();
32
33             _window = new Window("Adventure Time", 850, 400);
34
35             _gameStarted = false;
36             _gameOver = false;
37             _showingInstructions = false;
38
39             _swordUnlockedNotification = new SwordUnlocked();
40             _fastMovementNotification = new FastMovementUnlocked();
41             _highJumpNotification = new HighJumpUnlocked();
42
43             InitializeMap();
44             InitializePlayer();
45         }
46
47         private void GetUserName()
48         {
49             do
```

```
50             {
51                 Console.Write("Player name: ");
52                 _playerName = Console.ReadLine();
53                 if (string.IsNullOrWhiteSpace(_playerName))
54                 {
55                     Console.WriteLine("Invalid Name!!!");
56                 }
57             }
58             while (string.IsNullOrWhiteSpace(_playerName));
59         }
60
61     private void GetUserDescription()
62     {
63         do
64         {
65             Console.Write("Player description: ");
66             _playerDescription = Console.ReadLine();
67             if (string.IsNullOrWhiteSpace(_playerDescription))
68             {
69                 Console.WriteLine("Invalid Description!!!");
70             }
71         }
72         while (string.IsNullOrWhiteSpace(_playerDescription));
73     }
74
75     private void InitializeMap()
76     {
77         _map = new GameMap(new string[] { "jungle", "Jungle!", "You are in the jungle!" });
78         _map.SetUpGameMap();
79     }
80
81     private void InitializePlayer()
82     {
83         _player = new Player(_map, _playerName, _playerDescription);
84         _player.Attach(new ScoreObserver());
85         _player.Attach(new SwordObserver());
86         _player.Attach(_swordUnlockedNotification);
87     }
88
89     public void Run()
90     {
91         while (!_window.CloseRequested)
92         {
93             SplashKit.ProcessEvents();
94             _window.Clear(Color.White);
95
96             if (!_gameStarted)
97             {
```

```
98             HandleGameStart();
99         }
100        else if (_gameOver)
101    {
102        HandleGameOver();
103    }
104    else
105    {
106        HandleGamePlay();
107    }
108
109    SplashKit.RefreshScreen(60);
110}
111}
112
113 private void HandleGameStart()
114{
115    if (!_showingInstructions)
116    {
117        _interface = SplashKit.LoadBitmap("game start",
118                                         "GameStart.png");
119        SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
120        if (SplashKit.KeyTyped(KeyCode.SpaceKey))
121        {
122            _showingInstructions = true;
123        }
124    }
125    else
126    {
127        _interface = SplashKit.LoadBitmap("instructions",
128                                         "Instructions.png");
129        SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
130        if (SplashKit.KeyTyped(KeyCode.SpaceKey))
131        {
132            _gameStarted = true;
133        }
134    }
135}
136
137 private void HandleGameOver()
138{
139    _interface = SplashKit.LoadBitmap("game over",
140                                     "GameOver.png");
141    SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
142    if (SplashKit.KeyTyped(KeyCode.RKey))
143    {
144        _gameStarted = true;
145        _gameOver = false;
```

```
144             InitializeMap();
145             InitializePlayer();
146         }
147         if (SplashKit.KeyTyped(KeyCode.QKey))
148         {
149             _window.Close();
150         }
151     }
152 }
153
154 private void HandleGamePlay()
155 {
156     _map.Draw();
157     _player.Update();
158     _player.Draw();
159     CheckDeathState();
160     UpdatePlayerState();
161     UpdateInventory();
162     OpenInventory();
163     UpdateGameMap();
164 }
165
166 private void CheckDeathState()
167 {
168     if (_player.IsDead && ClockSingleton.getInstance
169         ().GetElapsedTicks("deathframe") >= 800)
170     {
171         Console.WriteLine("You lose!!!\n");
172         _gameOver = true;
173     }
174 }
175
176 private void UpdateInventory()
177 {
178     int golds = _player.CollectedGolds;
179     int swords = _player.CollectedSwords;
180
181     _player.Inventory = new Inventory();
182
183     if (golds > 0)
184     {
185         Item goldItem = new Item(new string[] { "gold" }, $"box of {golds} golds", "Very Precious Things");
186         _player.Inventory.Put(goldItem);
187     }
188
189     if (swords > 0)
190     {
191         Item swordItem = new Item(new string[] { "sword" }, "Metal
```

```
1         Sword", "Melee Weapon!");
191     _player.Inventory.Put(swordItem);
192 }
193 }
194
195     private void OpenInventory()
196     {
197         if (SplashKit.KeyTyped(KeyCode.TabKey))
198         {
199             string message = new LookCommand().Execute(_player, new
200                 string[] { "look at inventory" });
201             Console.WriteLine(message);
202         }
203     }
204
205     private void UpdateGameMap()
206     {
207         var keyToBackgroundMap = new Dictionary<KeyCode, (Bitmap,
208             string, string)>
209         {
210             { KeyCode.Num2Key, (_map.Sea, "You have moved to the
211                 sea!", "the sea") },
212             { KeyCode.Num1Key, (_map.Jungle, "You have moved to the
213                 jungle!", "the jungle") },
214             { KeyCode.Num3Key, (_map.Sky, "You have moved to the
215                 sky!", "the sky") },
216             { KeyCode.Num4Key, (_map.Desert, "You have moved to the
217                 desert!", "the desert") }
218         };
219
220         foreach (var entry in keyToBackgroundMap)
221         {
222             if (SplashKit.KeyTyped(entry.Key))
223             {
224                 if (_map.Background == entry.Value.Item1)
225                 {
226                     Console.WriteLine($"You are already in
{entry.Value.Item3}!");
227                 }
228                 else
229                 {
230                     _map.Background = entry.Value.Item1;
231                     Console.WriteLine(entry.Value.Item2);
232                 }
233                 break; // No need to check other keys once we find a
234                     match
235             }
236         }
237     }
238 }
```

```
231
232     private void UpdatePlayerState()
233     {
234         if (_player.CollectedGolds == 5)
235         {
236             _player.SetCollectStrategy(new AdvancedCollect());
237             _player.Notify("SwordUnlocked");
238             _player.Detach(_swordUnlockedNotification);
239         }
240
241         if (_player.IncreaseSpeed)
242         {
243             // Ensure the potion observer is attached
244             _player.Attach(_fastMovementNotification);
245             _player.Notify("SpeedPotionCollected");
246             _player.Detach(_fastMovementNotification);
247             _player.IncreaseSpeed = false; // Prevent repeated
248                                         notifications
249         }
250
251         if (_player.IncreaseJump)
252         {
253             _player.Attach(_highJumpNotification);
254             _player.Notify("JumpPotionCollected");
255             _player.Detach(_highJumpNotification);
256             _player.IncreaseJump = false; // Prevent repeated
257                                         notifications
258
259             // Check if the fast movement period is over
260             if (ClockSingleton.getInstance().GetElapsedTicks
261                 ("fastmovement") >= 3000)
262             {
263                 _player.SetMovementStrategy(new NormalMovement());
264
265                 // Check if the high jump period is over
266                 if (ClockSingleton.getInstance().GetElapsedTicks("highjump")
267                     >= 3000)
268                 {
269                     _player.SetJumpStrategy(new NormalJump());
270                 }
271 }
```