

```
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.IsNullOrEmpty(_playerName))
54             {
55                 Console.WriteLine("Invalid Name!!");
56             }
57         }
58         while (string.IsNullOrEmpty(_playerName));
59     }
60
61     private void GetUserDescription()
62     {
63         do
64         {
65             Console.Write("Player description: ");
66             _playerDescription = Console.ReadLine();
67             if (string.IsNullOrEmpty(_playerDescription))
68             {
69                 Console.WriteLine("Invalid Description!!");
70             }
71         }
72         while (string.IsNullOrEmpty(_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             {
```

```
198         HandleGameStart();
199     }
200     else if (_gameOver)
201     {
202         HandleGameOver();
203     }
204     else
205     {
206         HandleGamePlay();
207     }
208
209     SplashKit.RefreshScreen(60);
210 }
211 }
212
213 private void HandleGameStart()
214 {
215     if (!_showingInstructions)
216     {
217         _interface = SplashKit.LoadBitmap("game start",
218                                           "GameStart.png");
219         SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
220         if (SplashKit.KeyTyped(KeyCode.SpaceKey))
221         {
222             _showingInstructions = true;
223         }
224     }
225     else
226     {
227         _interface = SplashKit.LoadBitmap("instructions",
228                                           "Instructions.png");
229         SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
230         if (SplashKit.KeyTyped(KeyCode.SpaceKey))
231         {
232             _gameStarted = true;
233         }
234     }
235 }
236
237 private void HandleGameOver()
238 {
239     _interface = SplashKit.LoadBitmap("game over",
240                                       "GameOver.png");
241     SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
242     {
243         if (SplashKit.KeyTyped(KeyCode.RKey))
244         {
245             _gameStarted = true;
246             _gameOver = false;
247         }
248     }
249 }
```

```
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.CollectGolds;
179     int swords = _player.CollectSwords;
180
181     _player.Inventory = new Inventory();
182
183     if (golds > 0)
184     {
185         Item goldItem = new Item(new string[] { "gold" }, $"box of
186             {golds} golds", "Very Precious Things");
187         _player.Inventory.Put(goldItem);
188     }
189
190     if (swords > 0)
191     {
192         Item swordItem = new Item(new string[] { "sword" }, "Metal
```

```
        Sword", "Melee Weapon!");
        _player.Inventory.Put(swordItem);
    }
}

private void OpenInventory()
{
    if (SplashKit.KeyTyped(KeyCode.TabKey))
    {
        string message = new LookCommand().Execute(_player, new
            string[] { "look at inventory" });
        Console.WriteLine(message);
    }
}

private void UpdateGameMap()
{
    var keyToBackgroundMap = new Dictionary<KeyCode, (Bitmap,
        string, string)>
    {
        { KeyCode.Num2Key, (_map.Sea, "You have moved to the
            sea!", "the sea") },
        { KeyCode.Num1Key, (_map.Jungle, "You have moved to the
            jungle!", "the jungle") },
        { KeyCode.Num3Key, (_map.Sky, "You have moved to the
            sky!", "the sky") },
        { KeyCode.Num4Key, (_map.Desert, "You have moved to the
            desert!", "the desert") }
    };

    foreach (var entry in keyToBackgroundMap)
    {
        if (SplashKit.KeyTyped(entry.Key))
        {
            if (_map.Background == entry.Value.Item1)
            {
                Console.WriteLine($"You are already in
                    {entry.Value.Item3}!");
            }
            else
            {
                _map.Background = entry.Value.Item1;
                Console.WriteLine(entry.Value.Item2);
            }
            break; // No need to check other keys once we find a
                match
        }
    }
}
```

```
231
232     private void UpdatePlayerState()
233     {
234         if (_player.CollectGold == 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 notifications
248         }
249
250         if (_player.IncreaseJump)
251         {
252             _player.Attach(_highJumpNotification);
253             _player.Notify("JumpPotionCollected");
254             _player.Detach(_highJumpNotification);
255             _player.IncreaseJump = false; // Prevent repeated notifications
256         }
257
258         // Check if the fast movement period is over
259         if (ClockSingleton.GetInstance().GetElapsedTicks("fastmovement") >= 3000)
260         {
261             _player.SetMovementStrategy(new NormalMovement());
262         }
263
264         // Check if the high jump period is over
265         if (ClockSingleton.GetInstance().GetElapsedTicks("highjump") >= 3000)
266         {
267             _player.SetJumpStrategy(new NormalJump());
268         }
269     }
270 }
271 }
```