GameMap.cs


```
 1  using CustomProject.GameObjects;
 2  using SplashKitSDK;
 3
 4  namespace CustomProject
 5  {
 6      public class GameMap : GameObject
 7      {
 8          // Game map height and width
 9          private int _mapWidth, _mapHeight;
10
11          // Game map objects
12          private Bitmap _background;
13          private Bitmap _jungle, _sky, _sea, _desert;
14
15          private List<Land> _lands;
16          private List<CollectibleBomb> _bombs;
17          private List<CollectibleGold> _golds;
18          private List<Sword> _sword;
19          private List<SpeedPotion> _speedPotion;
20          private List<JumpPotion> _jumpPotion;
21
22          // For reading maps and draw
23          private string[] _txtMap;
24
25          public GameMap(string[] ids, string name, string description) :
              base(ids, name, description)
26          {
27              _lands = new List<Land>();
28              _bombs = new List<CollectibleBomb>();
29              _golds = new List<CollectibleGold>();
30              _sword = new List<Sword>();
31              _speedPotion = new List<SpeedPotion>();
32              _jumpPotion = new List<JumpPotion>();
33
34              ReadMapFromFile("map.txt");
35              LoadBackgrounds();
36          }
37
38          private void ReadMapFromFile(string filename)
39          {
40              StreamReader _rd = new StreamReader(filename);
41              _txtMap = _rd.ReadToEnd().Split(new[] { '\r', '\n' },
                  StringSplitOptions.RemoveEmptyEntries);
42              _rd.Close();
43
44              _mapHeight = _txtMap.Length;
45              _mapWidth = _txtMap[0].Length;
46          }
47
```

```
48          private void LoadBackgrounds()
49          {
50              _jungle = SplashKit.LoadBitmap("jungle", "jungle.png");
51              _sky = SplashKit.LoadBitmap("sky", "sky.png");
52              _sea = SplashKit.LoadBitmap("sea", "marine.png");
53              _desert = SplashKit.LoadBitmap("desert", "desert.jpg");
54              _background = _jungle;
55          }
56
57          // Set up the 2D array of the text map
58          public void SetUpGameMap()
59          {
60              for (int i = 0; i < _mapHeight; i++)
61              {
62                  for (int j = 0; j < _mapWidth; j++)
63                  {
64                      switch (_txtMap[i][j])
65                      {
66                          case '#':
67                              _lands.Add(new Land(j * 50, i * 50));
68                              break;
69                          case 'x':
70                              _bombs.Add(new CollectibleBomb(j * 50, i *
                        50));
71                              break;
72                          case 'o':
73                              _golds.Add(new CollectibleGold(j * 50, i *
                        50));
74                              break;
75                          case 's':
76                              _sword.Add(new Sword(j * 50, i * 50));
77                              break;
78                          case 'a':
79                              _speedPotion.Add(new SpeedPotion(j * 50, i *
                        50));
80                              break;
81                          case 'j':
82                              _jumpPotion.Add(new JumpPotion(j * 50, i *
                        50));
83                              break;
84                          default:
85                              break;
86                      }
87                  }
88              }
89          }
90
91          // Draw the game map using the 2D arry of text map
92          public void Draw()
```

```
93          {
94              SplashKit.DrawBitmap(_background, Camera.X, Camera.Y);
95              foreach (Land land in _lands)
96              {
97                  land.Draw();
98              }
99
100             foreach (CollectibleGold gold in _golds)
101             {
102                 gold.Draw();
103             }
104
105             foreach (CollectibleBomb bomb in _bombs)
106             {
107                 bomb.Draw();
108             }
109
110             foreach (Sword sword in _sword)
111             {
112                 sword.Draw();
113             }
114
115             foreach (SpeedPotion potion in _speedPotion)
116             {
117                 potion.Draw();
118             }
119
120             foreach (JumpPotion potion in _jumpPotion)
121             {
122                 potion.Draw();
123             }
124         }
125
126         public int Width { get => _mapWidth * 50; }
127         public int Height { get => _mapHeight * 50; }
128         public List<Land> Lands { get => _lands; }
129         public List<CollectibleBomb> Bombs { get => _bombs; }
130         public List<CollectibleGold> Golds { get => _golds; }
131         public List<Sword> Swords { get => _sword; }
132         public List<SpeedPotion> SpeedPotion { get => _speedPotion; }
133         public List<JumpPotion> JumpPotion { get => _jumpPotion; }
134         public Bitmap Background { get => _background; set => _background
              = value; }
135         public Bitmap Sea { get => _sea; }
136         public Bitmap Jungle { get => _jungle; }
137         public Bitmap Sky { get => _sky; }
138         public Bitmap Desert { get => _desert; }
139     }
140 }
```

GameProcessor.cs


```
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",
                       "GameStart.png");
118                  SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
119                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
120                  {
121                      _showingInstructions = true;
122                  }
123              }
124              else
125              {
126                  _interface = SplashKit.LoadBitmap("instructions",
                       "Instructions.png");
127                  SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
128                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
129                  {
130                      _gameStarted = true;
131                  }
132              }
133          }
134
135          private void HandleGameOver()
136          {
137              _interface = SplashKit.LoadBitmap("game over",
                    "GameOver.png");
138              SplashKit.DrawBitmap(_interface, Camera.X, Camera.Y);
139              {
140                  if (SplashKit.KeyTyped(KeyCode.RKey))
141                  {
142                      _gameStarted = true;
143                      _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
                  ().GetElapsedTicks("deathframe") >= 800)
169             {
170                 Console.WriteLine("You lose!!!\n");
171                 _gameOver = true;
172             }
173         }
174 
175         private void UpdateInventory()
176         {
177             int golds = _player.CollectedGolds;
178             int swords = _player.CollectedSwords;
179 
180             _player.Inventory = new Inventory();
181 
182             if (golds > 0)
183             {
184                 Item goldItem = new Item(new string[] { "gold" }, $"box of
                      {golds} golds", "Very Precious Things");
185                 _player.Inventory.Put(goldItem);
186             }
187 
188             if (swords > 0)
189             {
190                 Item swordItem = new Item(new string[] { "sword" }, "Metal
```

```
                    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
                      string[] { "look at inventory" });
200                 Console.WriteLine(message);
201             }
202         }
203
204         private void UpdateGameMap()
205         {
206             var keyToBackgroundMap = new Dictionary<KeyCode, (Bitmap,
                  string, string)>
207             {
208                 { KeyCode.Num2Key, (_map.Sea, "You have moved to the
                      sea!", "the sea") },
209                 { KeyCode.Num1Key, (_map.Jungle, "You have moved to the
                      jungle!", "the jungle") },
210                 { KeyCode.Num3Key, (_map.Sky, "You have moved to the
                      sky!", "the sky") } ,
211                 { KeyCode.Num4Key, (_map.Desert, "You have moved to the
                      desert!", "the desert") }
212             };
213
214             foreach (var entry in keyToBackgroundMap)
215             {
216                 if (SplashKit.KeyTyped(entry.Key))
217                 {
218                     if (_map.Background == entry.Value.Item1)
219                     {
220                         Console.WriteLine($"You are already in
                        {entry.Value.Item3}!");
221                     }
222                     else
223                     {
224                         _map.Background = entry.Value.Item1;
225                         Console.WriteLine(entry.Value.Item2);
226                     }
227                     break; // No need to check other keys once we find a
                        match
228                 }
229             }
230         }
```

```
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
                      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 }
```

IdentifiableObject.cs


```
 1  namespace CustomProject
 2  {
 3      public class IdentifiableObject
 4      {
 5          private List<string> _identifiers = new List<string>();
 6
 7          public IdentifiableObject(string[] idents)
 8          {
 9              for (int i = 0; i < idents.Length; i++)
10              {
11                  _identifiers.Add(idents[i].ToLower());
12              }
13          }
14          public bool AreYou(string id)
15          {
16              if (_identifiers.Contains(id.ToLower()))
17              {
18                  return true;
19              }
20              return false;
21          }
22          public string FirstId()
23          {
24              if (_identifiers.Any())
25              {
26                  return _identifiers.First();
27              }
28              return "";
29          }
30      }
31  }
32
```

Item.cs


```
1  using CustomProject.GameObjects;
2
3  namespace CustomProject
4  {
5      public class Item : GameObject
6      {
7          public Item(string[] idents, string name, string description) :
             base(idents, name, description)
8          {
9          }
10     }
11 }
12
```

Inventory.cs


```
1  namespace CustomProject
2  {
3      public class Inventory
4      {
5          private List<Item> _items;
6
7          public Inventory()
8          {
9              _items = new List<Item>();
10         }
11
12         public bool HasItem(string id)
13         {
14             foreach (Item itm in _items)
15             {
16                 if (itm.AreYou(id))
17                 {
18                     return true;
19                 }
20             }
21             return false;
22         }
23
24         public void Put(Item itm)
25         {
26             _items.Add(itm);
27         }
28
29         public Item Fetch(string id)
30         {
31             foreach (Item itm in _items)
32             {
33                 if (itm.AreYou(id))
34                 {
35                     return itm;
36                 }
37             }
38             return null;
39         }
40
41         public string ItemList
42         {
43             get
44             {
45                 string list = "";
46                 foreach (Item itm in _items)
47                 {
48                     list += "    " + itm.ShortDescription;
49                 }
```

```
50                  return list;
51              }
52          }
53      }
54  }
55
```

IHaveInventory.cs


```
1  using CustomProject.GameObjects;
2
3  namespace CustomProject
4  {
5      interface IHaveInventory
6      {
7          GameObject Locate(string id);
8
9          string Name
10         {
11             get => Name;
12         }
13     }
14 }
15
```

Player.cs


```
 1  using CustomProject.GameHandler;
 2  using CustomProject.GameObjects;
 3  using CustomProject.Observers;
 4  using CustomProject.StrategyDesign;
 5  using SplashKitSDK;
 6
 7  namespace CustomProject
 8  {
 9      public class Player : DrawableObject, IHaveInventory, ISubject
10      {
11          private double _gravity;
12          private AnimationScript _playerScript;
13          private Sprite _playerSprite;
14          private GameMap _gameMap;
15          private bool _isDead;
16          private bool _isMoving;
17
18          private Inventory _inventory;
19          private int _collectedGolds;
20          private int _collectedSwords;
21          private bool _increaseSpeed;
22          private bool _increaseJump;
23
24          private IMovementStrategy _movementStrategy;
25          private IJumpStrategy _jumpStrategy;
26          private ICollectStrategy _collectStrategy;
27
28          private List<IObserver> _observers;
29
30          private AnimationHandler _animationHandler;
31          private CameraHandler _cameraHandler;
32          private CollisionHandler _collisionHandler;
33
34          public Player(GameMap gameMap, string name, string description) :
              base(200, 200, new string[] { "inventory" }, name, description,
              "playerImage.png")
35          {
36              InitializePlayerSprite();
37              InitializePlayerState(gameMap);
38              InitializeStrategy();
39
40              _animationHandler = new AnimationHandler(this);
41              _cameraHandler = new CameraHandler(gameMap, this);
42              _collisionHandler = new CollisionHandler(gameMap, this);
43
44              _observers = new List<IObserver>();
45          }
46
47          // Update the actions of the player
```

```
48          public void Update()
49          {
50              if (_isDead)
51              {
52                  _animationHandler.UpdateDeadAnimation();
53              }
54
55              _movementStrategy.Move(this);
56              _collectStrategy.Collect(this);
57              UpdateYLocation();
58
59              _animationHandler.UpdateMovingAnimation();
60              _cameraHandler.HandleCamera();
61              _playerSprite.UpdateAnimation();
62              _collisionHandler.CheckBombCollision();
63          }
64
65          private void UpdateYLocation()
66          {
67              if (!_isDead) // Check for jumping keypressed
68              {
69                  _jumpStrategy.Jump(this);
70              }
71
72              // Update the gravity and Y location
73              _gravity += 1;
74              double newY = _yLocation + _gravity;
75
76              if (_gravity > 0 && _collisionHandler.IsCollideWithLand
                  (_xLocation, newY))
77              {
78                  _gravity = 0;
79              }
80              else if (_gravity < 0 && _collisionHandler.IsCollideWithLand
                  (_xLocation, newY))
81              {
82                  _gravity = 0;
83              }
84              else
85              {
86                  _yLocation = newY;
87              }
88          }
89
90          // Draw the player on the screen
91          public override void Draw()
92          {
93              _playerSprite.Draw(_xLocation, _yLocation);
94          }
```

```
 95
 96          private void InitializePlayerSprite()
 97          {
 98              _image.SetCellDetails(_image.Width / 8, _image.Height / 6, 8,
                   6, 48);
 99              _playerScript = SplashKit.LoadAnimationScript("_playerScript",
                    "animationscript.txt");
100              _playerSprite = SplashKit.CreateSprite(_image, _playerScript);
101              _playerSprite.StartAnimation(0);
102          }
103
104          private void InitializePlayerState(GameMap gameMap)
105          {
106              _gravity = 0;
107              _isDead = false;
108              _isMoving = false;
109              _gameMap = gameMap;
110              _inventory = new Inventory();
111              _collectedGolds = 0;
112              _collectedSwords = 0;
113              _increaseSpeed = false;
114              _increaseJump = false;
115          }
116
117          private void InitializeStrategy()
118          {
119              _movementStrategy = new NormalMovement();
120              _jumpStrategy = new NormalJump();
121              _collectStrategy = new NormalCollect();
122          }
123
124          public void Attach(IObserver observer)
125          {
126              _observers.Add(observer);
127          }
128
129          public void Detach(IObserver observer)
130          {
131              _observers.Remove(observer);
132          }
133
134          public void Notify(string eventType)
135          {
136              foreach (IObserver observer in _observers)
137              {
138                  observer.Update(eventType, this);
139              }
140          }
141
```

```
142         public GameObject Locate(string id)
143         {
144             if (AreYou(id))
145             {
146                 return this;
147             }
148             else if (_inventory.HasItem(id))
149             {
150                 return _inventory.Fetch(id);
151             }
152             else
153             {
154                 return null;
155             }
156         }
157 
158         public void SetMovementStrategy(IMovementStrategy strategy)
159         {
160             _movementStrategy = strategy;
161         }
162 
163         public void SetJumpStrategy(IJumpStrategy strategy)
164         {
165             _jumpStrategy = strategy;
166         }
167 
168         public void SetCollectStrategy(ICollectStrategy strategy)
169         {
170             _collectStrategy = strategy;
171         }
172 
173         public override string FullDescription
174         {
175             get
176             {
177                 string fulldesc = "\n" + "-----INVENTORY-----" + "\n";
178                 fulldesc += $"You are {Name}, {base.FullDescription}\n";
179                 fulldesc += "You are carrying\n";
180                 fulldesc += $"{_inventory.ItemList}";
181                 return fulldesc;
182             }
183         }
184 
185         public Inventory Inventory
186         {
187             get => _inventory;
188             set => _inventory = value;
189         }
190 
```

```
191         public double Gravity { get => _gravity; set => _gravity =
              value; }
192         public bool IsMoving { get => _isMoving; set => _isMoving =
              value; }
193         public bool IsDead { get => _isDead; set => _isDead = value; }
194         public GameMap GameMap { get => _gameMap; }
195         public int CollectedGolds { get => _collectedGolds; set =>
              _collectedGolds = value; }
196         public int CollectedSwords { get => _collectedSwords; set =>
              _collectedSwords = value; }
197         public bool IncreaseSpeed { get => _increaseSpeed; set =>
              _increaseSpeed = value; }
198         public bool IncreaseJump { get => _increaseJump; set =>
              _increaseJump = value; }
199         public Sprite PlayerSprite { get => _playerSprite; }
200         public CollisionHandler CollisionHandler { get =>
              _collisionHandler; }
201     }
202 }
203
204
```

Program.cs


```
 1  namespace CustomProject
 2  {
 3      class Program
 4      {
 5          public static void Main()
 6          {
 7              new GameProcessor().Run();
 8          }
 9      }
10  }
11
```

Command.cs


```
1  namespace CustomProject.Command
2  {
3      public abstract class Command : IdentifiableObject
4      {
5          public Command(string[] ids) : base(new string[] { "command" })
6          {
7
8          }
9
10         public abstract string Execute(Player p, string[] text);
11     }
12 }
13
```

LookCommand.cs


```
1  namespace CustomProject.Command
2  {
3      public class LookCommand : Command
4      {
5          public LookCommand() : base(new string[] { "look" })
6          {
7
8          }
9
10         public override string Execute(Player p, string[] text)
11         {
12             string thingId;
13             string thing;
14             Item item;
15             string[] array = text[0].Split(" ");
16
17             thingId = array[2];
18             thing = LookAtIn(thingId, p);
19             return $"{thing}\n";
20         }
21
22         private string LookAtIn(string thingId, IHaveInventory container)
23         {
24             if (container.Locate(thingId) == null)
25             {
26                 return null;
27             }
28             return container.Locate(thingId).FullDescription;
29         }
30     }
31 }
```

AnimationHandler.cs


```
1  using CustomProject.SingletonDesign;
2  using SplashKitSDK;
3
4  namespace CustomProject.GameHandler
5  {
6      public class AnimationHandler
7      {
8          private Player _player;
9
10         public AnimationHandler(Player player)
11         {
12             _player = player;
13         }
14
15         public void UpdateMovingAnimation()
16         {
17             if (_player.IsDead) return;
18
19             if (_player.IsMoving)
20             {
21                 UpdateRunningAnimation();
22             }
23             else
24             {
25                 UpdateStandingAnimation();
26             }
27         }
28
29         private void UpdateStandingAnimation()
30         {
31             if (_player.PlayerSprite.AnimationName() == "runleft")
32             {
33                 _player.PlayerSprite.StartAnimation("Left");
34             }
35             else if (_player.PlayerSprite.AnimationName() == "runright")
36             {
37                 _player.PlayerSprite.StartAnimation("Right");
38             }
39         }
40
41         private void UpdateRunningAnimation()
42         {
43             if (SplashKit.KeyDown(KeyCode.LeftKey))
44             {
45                 StartAnimationIfNotRunning("RunLeft");
46             }
47             else if (SplashKit.KeyDown(KeyCode.RightKey))
48             {
49                 StartAnimationIfNotRunning("RunRight");
```

```
50              }
51          }
52
53          private void StartAnimationIfNotRunning(string animationName)
54          {
55              if (SplashKit.SpriteAnimationName(_player.PlayerSprite) !=
                  animationName.ToLower())
56              {
57                  _player.PlayerSprite.StartAnimation(animationName);
58              }
59          }
60
61          public void UpdateDeadAnimation()
62          {
63              if (ClockSingleton.getInstance().GetElapsedTicks("deathframe")
                  <= 800)
64              {
65                  _player.PlayerSprite.UpdateAnimation();
66              }
67          }
68
69          public void Die()
70          {
71              _player.IsDead = true;
72              ClockSingleton.getInstance().StartTimer("deathframe");
73              if (_player.PlayerSprite.AnimationName().Contains("right"))
74              {
75                  _player.PlayerSprite.StartAnimation("DeadRight");
76              }
77              else if (_player.PlayerSprite.AnimationName().Contains("left"))
78              {
79                  _player.PlayerSprite.StartAnimation("DeadLeft");
80              }
81          }
82      }
83  }
84
```

CameraHandler.cs


```
1  using SplashKitSDK;
2
3  namespace CustomProject.GameHandler
4  {
5      public class CameraHandler
6      {
7          private GameMap _gameMap;
8          private Player _player;
9
10         public CameraHandler(GameMap gamemap, Player player)
11         {
12             _player = player;
13             _gameMap = gamemap;
14         }
15
16         public void HandleCamera()
17         {
18             // Initialize the fixed camera position on the screen
19             Camera.X = _player.X - SplashKit.ScreenWidth() / 2;
20             Camera.Y = _player.Y - SplashKit.ScreenHeight() / 2;
21
22             // Update the camera as the player moves
23             if (Camera.X < 0)
24             {
25                 Camera.X = 0;
26             }
27             if (Camera.Y < 0)
28             {
29                 Camera.Y = 0;
30             }
31             if (Camera.X > _gameMap.Width - SplashKit.ScreenWidth())
32             {
33                 Camera.X = _gameMap.Width - SplashKit.ScreenWidth();
34             }
35             if (Camera.Y > _gameMap.Height - SplashKit.ScreenHeight())
36             {
37                 Camera.Y = _gameMap.Height - SplashKit.ScreenHeight();
38             }
39         }
40     }
41 }
42
```

CollisionHandler.cs


```
1  using CustomProject.GameObjects;
2  using SplashKitSDK;
3
4  namespace CustomProject.GameHandler
5  {
6      public class CollisionHandler
7      {
8          private GameMap _gameMap;
9          private Player _player;
10
11         public CollisionHandler(GameMap gamemap, Player player)
12         {
13             _gameMap = gamemap;
14             _player = player;
15         }
16
17         // Process the action of touching bombs and remove them
18         public void CheckBombCollision()
19         {
20             List<CollectibleBomb> removedBombs = new List<CollectibleBomb>
                 ();
21             AnimationHandler _animationHandler = new AnimationHandler
                 (_player);
22             foreach (CollectibleBomb bomb in _gameMap.Bombs)
23             {
24                 if (SplashKit.BitmapCollision(_player.Image, _player.X,
                     _player.Y, bomb.Image, bomb.X, bomb.Y))
25                 {
26                     removedBombs.Add(bomb);
27                     _player.IsDead = true;
28                     _animationHandler.Die();
29                 }
30             }
31
32             foreach (CollectibleBomb bomb in removedBombs)
33             {
34                 _gameMap.Bombs.Remove(bomb);
35             }
36         }
37
38         // Check if the player is on the ground
39         public bool IsPlayerOnGround()
40         {
41             return IsCollideWithLand(_player.X, _player.Y + 1);
42         }
43
44         // Check land collision with player for handle movement
45         public bool IsCollideWithLand(double x, double y)
46         {
```

```
47              foreach (Land land in _gameMap.Lands)
48              {
49                  if (SplashKit.BitmapCollision(_player.Image, x, y,
                      land.Image, land.X, land.Y))
50                  {
51                      return true;
52                  }
53              }
54              return false;
55          }
56      }
57  }
58
```

GameObject.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public abstract class GameObject : IdentifiableObject
4      {
5          private string _description;
6          private string _name;
7
8          public GameObject(string[] ids, string name, string description) :
             base(ids)
9          {
10             _description = description;
11             _name = name;
12         }
13
14         public string Name
15         {
16             get => _name;
17         }
18
19         public string ShortDescription
20         {
21             get => $"a {Name} ({FirstId()})\n";
22         }
23
24         public virtual string FullDescription
25         {
26             get => _description;
27         }
28     }
29 }
30
```

DrawableObject.cs


```
1  using SplashKitSDK;
2
3  namespace CustomProject.GameObjects
4  {
5      public abstract class DrawableObject : GameObject
6      {
7          protected double _xLocation;
8          protected double _yLocation;
9          protected Bitmap _image;
10
11         public DrawableObject(double xLocation, double yLocation, string[]
             ids, string name, string description, string imagePath)
12             : base(ids, name, description)
13         {
14             _xLocation = xLocation;
15             _yLocation = yLocation;
16             _image = SplashKit.LoadBitmap(ids[0], imagePath);
17         }
18
19         public virtual void Draw()
20         {
21             SplashKit.DrawBitmap(_image, _xLocation, _yLocation);
22         }
23
24         public double X { get => _xLocation; set => _xLocation = value; }
25         public double Y { get => _yLocation; set => _yLocation = value; }
26         public Bitmap Image => _image;
27     }
28 }
29
```

Bomb.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class CollectibleBomb : DrawableObject
4      {
5          public CollectibleBomb(double xLocation, double yLocation) : base
             (xLocation, yLocation, new string[] { "bomb" }, "Bombs!!", "Do
             not touch the bombs!", "Bomb.png")
6          {
7          }
8      }
9  }
10
```

Gold.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class CollectibleGold : DrawableObject
4      {
5          public CollectibleGold(double xLocation, double yLocation) : base
             (xLocation, yLocation, new string[] { "gold" }, "Golds!!",
             "Collect golds to win the game!!", "Gold.png")
6          {
7          }
8      }
9  }
```

Land.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class Land : DrawableObject
4      {
5          public Land(double xLocation, double yLocation) : base(xLocation,
             yLocation, new string[] { "" }, "", "", "land.png")
6          {
7          }
8      }
9  }
```

Sword.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class Sword : DrawableObject
4      {
5          public Sword(double xLocation, double yLocation) : base(xLocation,
             yLocation, new string[] { "sword" }, "Metal Sword!", "Meelee
             weapons!", "sword.png")
6          {
7          }
8      }
9  }
```

SpeedPotion.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class SpeedPotion : DrawableObject
4      {
5          public SpeedPotion(double xLocation, double yLocation) : base
             (xLocation, yLocation, new string[] { "speed potion" }, "Speed
             Potion!!", "Increase speed !!", "speedpotion.png")
6          {
7          }
8      }
9  }
10
```

JumpPotion.cs


```
1  namespace CustomProject.GameObjects
2  {
3      public class JumpPotion : DrawableObject
4      {
5          public JumpPotion(double xLocation, double yLocation) : base
             (xLocation, yLocation, new string[] { "jump potion" }, "Jump
             Potion!!", "Increase jump !!", "jumppotion.png")
6          {
7          }
8      }
9  }
10
```

IObserver.cs


```
1  namespace CustomProject.Observers
2  {
3      public interface IObserver
4      {
5          void Update(string eventType, Player player);
6      }
7  }
8
```

ISubject.cs


```
1  namespace CustomProject.Observers
2  {
3      public interface ISubject
4      {
5          void Attach(IObserver observer);
6          void Detach(IObserver observer);
7          void Notify(string eventType);
8      }
9  }
10
```

FastMovementUnlocked.cs


```
 1 using CustomProject.SingletonDesign;
 2 using CustomProject.StrategyDesign;
 3 namespace CustomProject.Observers
 4 {
 5     public class FastMovementUnlocked : IObserver
 6     {
 7         public void Update(string eventType, Player player)
 8         {
 9             if (eventType == "SpeedPotionCollected")
10             {
11                 Console.WriteLine("Your speed has been increased for 3
                     seconds!");
12                 player.SetMovementStrategy(new FastMovement());
13                 ClockSingleton.getInstance().StartTimer("fastmovement");
14             }
15         }
16     }
17 }
18
```

HighJumpUnlocked.cs


```
 1 using CustomProject.SingletonDesign;
 2 using CustomProject.StrategyDesign;
 3 namespace CustomProject.Observers
 4 {
 5     public class HighJumpUnlocked : IObserver
 6     {
 7         public void Update(string eventType, Player player)
 8         {
 9             if (eventType == "JumpPotionCollected")
10             {
11                 Console.WriteLine("Your jump ability has been increased for ↪
                       3 seconds!");
12                 player.SetJumpStrategy(new HighJump());
13                 ClockSingleton.getInstance().StartTimer("highjump");
14             }
15         }
16     }
17 }
18
```

ScoreObserver.cs


```
1  namespace CustomProject.Observers
2  {
3      public class ScoreObserver : IObserver
4      {
5          private int _score;
6
7          public void Update(string eventType, Player player)
8          {
9              if (eventType == "GoldCollected")
10             {
11                 _score = player.CollectedGolds * 10;
12                 Console.WriteLine($"Score updated: {_score}");
13             }
14         }
15     }
16 }
17
```

SwordObserver.cs


```
1  namespace CustomProject.Observers
2  {
3      public class SwordObserver : IObserver
4      {
5          public void Update(string eventType, Player player)
6          {
7              if (eventType == "SwordCollected")
8              {
9                  Console.WriteLine($"Swords collected:
                     {player.CollectedSwords}");
10             }
11         }
12     }
13 }
14
```

SwordUnlocked.cs


```
1  namespace CustomProject.Observers
2  {
3      public class SwordUnlocked : IObserver
4      {
5          public void Update(string eventType, Player player)
6          {
7              if (eventType == "SwordUnlocked")
8              {
9                  Console.WriteLine($"\nYou are now able to collect Sword and ➥
                       the Potions!\n");
10             }
11         }
12     }
13 }
14
```

ClockSingleton.cs


```
1  namespace CustomProject.SingletonDesign
2  {
3      public class ClockSingleton
4      {
5          private static readonly object _lockObject = new object();
6          private static volatile ClockSingleton _instance;
7          private Dictionary<string, SplashKitSDK.Timer> _timers;
8
9          private ClockSingleton()
10         {
11             _timers = new Dictionary<string, SplashKitSDK.Timer>();
12         }
13
14         public static ClockSingleton getInstance()
15         {
16             if (_instance == null)
17             {
18                 lock (_lockObject)
19                 {
20                     if (_instance == null)
21                     {
22                         _instance = new ClockSingleton();
23                     }
24                 }
25             }
26             return _instance;
27         }
28
29         public void StartTimer(string timerName)
30         {
31             if (_timers.ContainsKey(timerName))
32             {
33                 _timers[timerName].Reset();
34             }
35             else
36             {
37                 _timers[timerName] = new SplashKitSDK.Timer(timerName);
38             }
39             _timers[timerName].Start();
40         }
41
42         public uint GetElapsedTicks(string timerName)
43         {
44             if (_timers.ContainsKey(timerName))
45             {
46                 return _timers[timerName].Ticks;
47             }
48             return 0;
49         }
```

```
50      }
51  }
52
```

IJumpStrategy.cs


```
1  namespace CustomProject
2  {
3      public interface IJumpStrategy
4      {
5          void Jump(Player player);
6      }
7  }
8
```

IMovementStrategy.cs


```
1  namespace CustomProject
2  {
3      public interface IMovementStrategy
4      {
5          void Move(Player player);
6      }
7  }
8
```

ICollectStrategy.cs


```
1  namespace CustomProject.StrategyDesign
2  {
3      public interface ICollectStrategy
4      {
5          void Collect(Player player);
6      }
7  }
8
```

NormalJump.cs


```
1  using SplashKitSDK;
2
3  namespace CustomProject.StrategyDesign
4  {
5      public class NormalJump : IJumpStrategy
6      {
7          public void Jump(Player player)
8          {
9              if (SplashKit.KeyTyped(KeyCode.UpKey))
10             {
11                 if (player.CollisionHandler.IsPlayerOnGround())
12                 {
13                     player.Gravity = -20;
14                 }
15             }
16         }
17     }
18 }
19
20
```

HighJump.cs


```
1  using SplashKitSDK;
2
3  namespace CustomProject.StrategyDesign
4  {
5      public class HighJump : IJumpStrategy
6      {
7          public void Jump(Player player)
8          {
9              if (SplashKit.KeyTyped(KeyCode.UpKey))
10             {
11                 if (player.CollisionHandler.IsPlayerOnGround())
12                 {
13                     player.Gravity = -25;
14                 }
15             }
16         }
17     }
18 }
19
```

NormalMovement.cs


```
1  using CustomProject.GameHandler;
2  using SplashKitSDK;
3
4  namespace CustomProject.StrategyDesign
5  {
6      public class NormalMovement : IMovementStrategy
7      {
8          public void Move(Player player)
9          {
10             player.IsMoving = false;
11             if (!player.IsDead)
12             {
13                 if (SplashKit.KeyDown(KeyCode.LeftKey))
14                 {
15                     player.IsMoving = true;
16                     if (!player.CollisionHandler.IsCollideWithLand(player.X
                       - 3, player.Y))
17                     {
18                         player.X -= 3;
19                     }
20                 }
21                 else if (SplashKit.KeyDown(KeyCode.RightKey))
22                 {
23                     player.IsMoving = true;
24                     if (!player.CollisionHandler.IsCollideWithLand(player.X
                       + 3, player.Y))
25                     {
26                         player.X += 3;
27                     }
28                 }
29             }
30         }
31     }
32 }
33
```

FastMovement.cs


```
1  using SplashKitSDK;
2
3  namespace CustomProject.StrategyDesign
4  {
5      public class FastMovement : IMovementStrategy
6      {
7          public void Move(Player player)
8          {
9              player.IsMoving = false;
10             if (!player.IsDead)
11             {
12                 if (SplashKit.KeyDown(KeyCode.LeftKey))
13                 {
14                     player.IsMoving = true;
15                     if (!player.CollisionHandler.IsCollideWithLand(player.X
                       - 6, player.Y))
16                     {
17                         player.X -= 6;
18                     }
19                 }
20                 else if (SplashKit.KeyDown(KeyCode.RightKey))
21                 {
22                     player.IsMoving = true;
23                     if (!player.CollisionHandler.IsCollideWithLand(player.X
                       + 6, player.Y))
24                     {
25                         player.X += 6;
26                     }
27                 }
28             }
29         }
30     }
31 }
32
```

NormalCollect.cs


```
1  using CustomProject.GameObjects;
2  using SplashKitSDK;
3
4  namespace CustomProject.StrategyDesign
5  {
6      public class NormalCollect : ICollectStrategy
7      {
8          public void Collect(Player player)
9          {
10             if (!player.IsDead)
11             {
12                 List<CollectibleGold> collectedGolds = new
                       List<CollectibleGold>();
13
14                 foreach (CollectibleGold gold in player.GameMap.Golds)
15                 {
16                     if (SplashKit.BitmapCollision(player.Image, player.X,
                       player.Y, gold.Image, gold.X, gold.Y))
17                     {
18                         collectedGolds.Add(gold);
19                     }
20                 }
21
22                 foreach (CollectibleGold gold in collectedGolds)
23                 {
24                     player.GameMap.Golds.Remove(gold);
25                     player.CollectedGolds += 1;
26                     player.Notify("GoldCollected");
27                 }
28             }
29         }
30     }
31 }
32
```

AdvancedCollect.cs

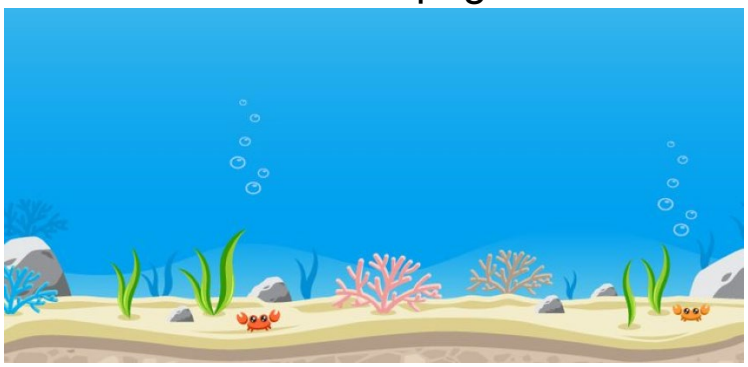

```
1  using CustomProject.GameObjects;
2  using SplashKitSDK;
3
4  namespace CustomProject.StrategyDesign
5  {
6      public class AdvancedCollect : ICollectStrategy
7      {
8          public void Collect(Player player)
9          {
10             if (!player.IsDead)
11             {
12                 List<CollectibleGold> collectedGolds = new
                     List<CollectibleGold>();
13                 List<Sword> collectedSwords = new List<Sword>();
14                 List<SpeedPotion> collectedSpeedPotion = new
                     List<SpeedPotion>();
15                 List<JumpPotion> collectedJumpPotion = new List<JumpPotion>
                     ();
16
17                 foreach (CollectibleGold gold in player.GameMap.Golds)
18                 {
19                     if (SplashKit.BitmapCollision(player.Image, player.X,
                       player.Y, gold.Image, gold.X, gold.Y))
20                     {
21                         collectedGolds.Add(gold);
22                     }
23                 }
24
25                 foreach (Sword sword in player.GameMap.Swords)
26                 {
27                     if (SplashKit.BitmapCollision(player.Image, player.X,
                       player.Y, sword.Image, sword.X, sword.Y))
28                     {
29                         collectedSwords.Add(sword);
30                     }
31                 }
32
33                 foreach (SpeedPotion potion in player.GameMap.SpeedPotion)
34                 {
35                     if (SplashKit.BitmapCollision(player.Image, player.X,
                       player.Y, potion.Image, potion.X, potion.Y))
36                     {
37                         collectedSpeedPotion.Add(potion);
38                     }
39                 }
40
41                 foreach (JumpPotion potion in player.GameMap.JumpPotion)
42                 {
43                     if (SplashKit.BitmapCollision(player.Image, player.X,
```

```
                        player.Y, potion.Image, potion.X, potion.Y))
44                      {
45                          collectedJumpPotion.Add(potion);
46                      }
47                  }
48
49                  foreach (SpeedPotion potion in collectedSpeedPotion)
50                  {
51                      player.GameMap.SpeedPotion.Remove(potion);
52                      player.IncreaseSpeed = true;
53                  }
54
55                  foreach (JumpPotion potion in collectedJumpPotion)
56                  {
57                      player.GameMap.JumpPotion.Remove(potion);
58                      player.IncreaseJump = true;
59                  }
60
61                  foreach (Sword sword in collectedSwords)
62                  {
63                      player.GameMap.Swords.Remove(sword);
64                      player.CollectedSwords += 1;
65                      player.Notify("SwordCollected");
66                  }
67
68                  foreach (CollectibleGold gold in collectedGolds)
69                  {
70                      player.GameMap.Golds.Remove(gold);
71                      player.CollectedGolds += 1;
72                      player.Notify("GoldCollected");
73                  }
74              }
75          }
76      }
77  }
78
79
```

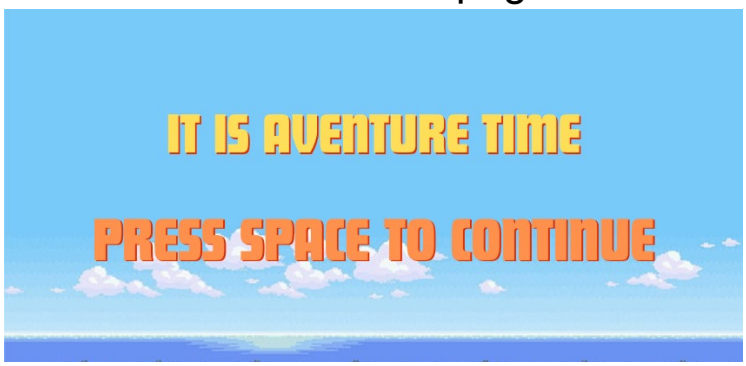
animationscript.txt

```
SplashKit Animation

// f: identifier, cell number, duration, next frame
f:[0-7],[0-7],12,0
f:[8-15],[8-15],12,8
f:[16-23],[23-16],8,16
f:[24-31],[24-31],8,24
f:[32-39],[32-39],12,32
f:[40-47],[47-40],12,40

// identifiers
i:Right,0
i:Left,8
i:RunLeft,16
i:RunRight,24
i:DeadRight,32
i:DeadLeft,40
```

marine.png

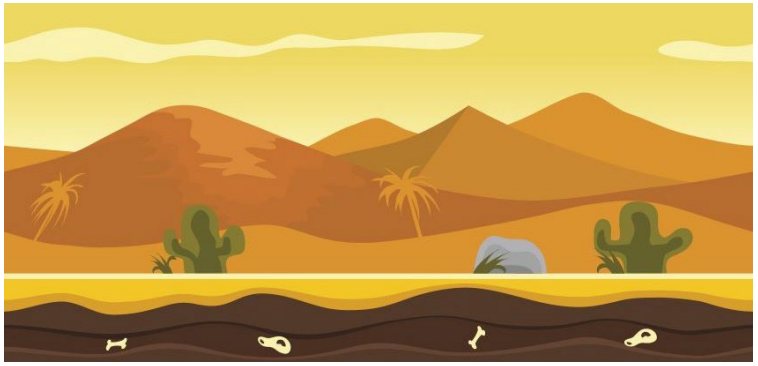


GameStart.png

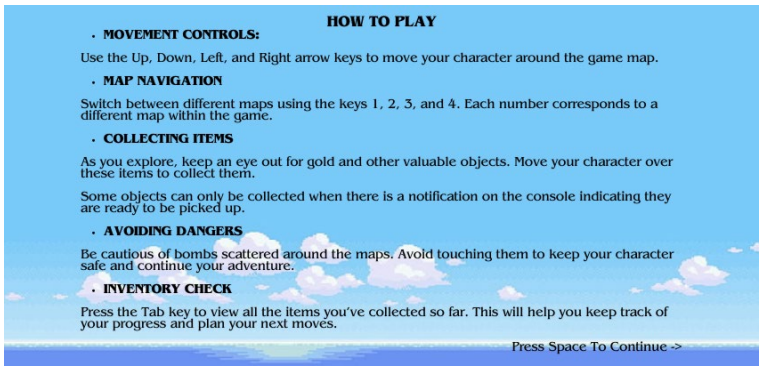


sky.png

desert.jpg



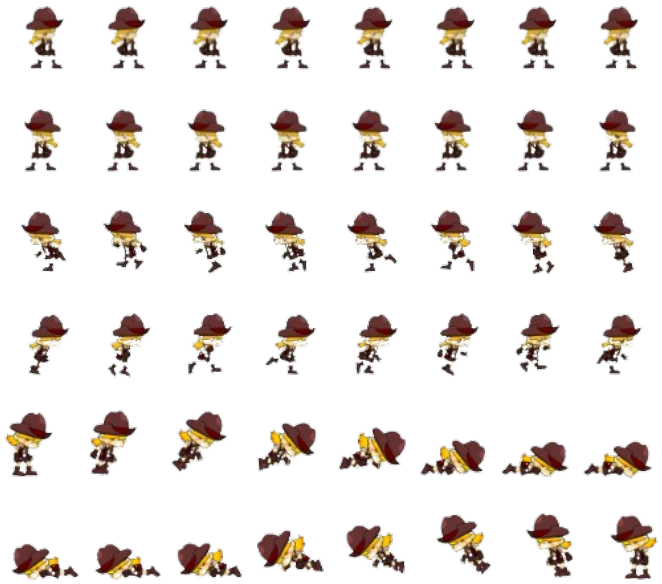
Instructions.png



jungle.png

GameOver.png

playerImage.png



jumppotion.png

speedpotion.png

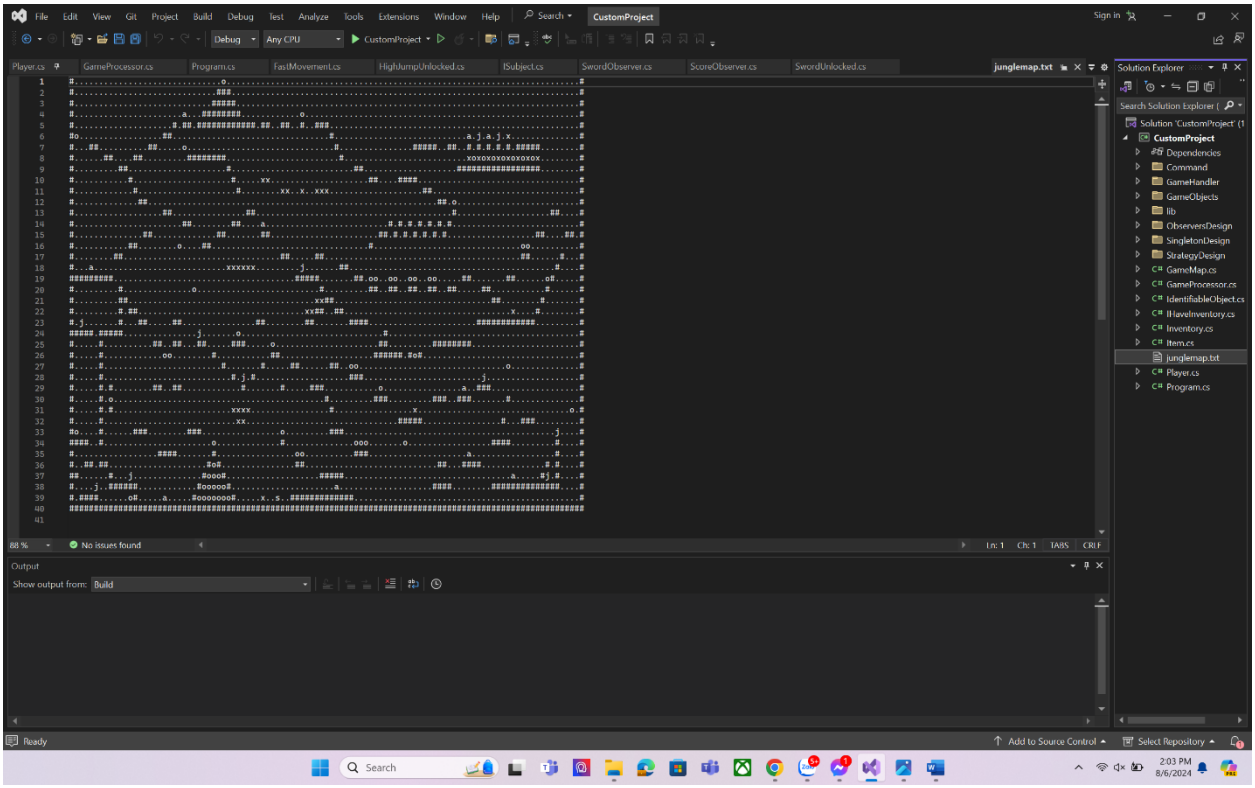
sword.png

land.png

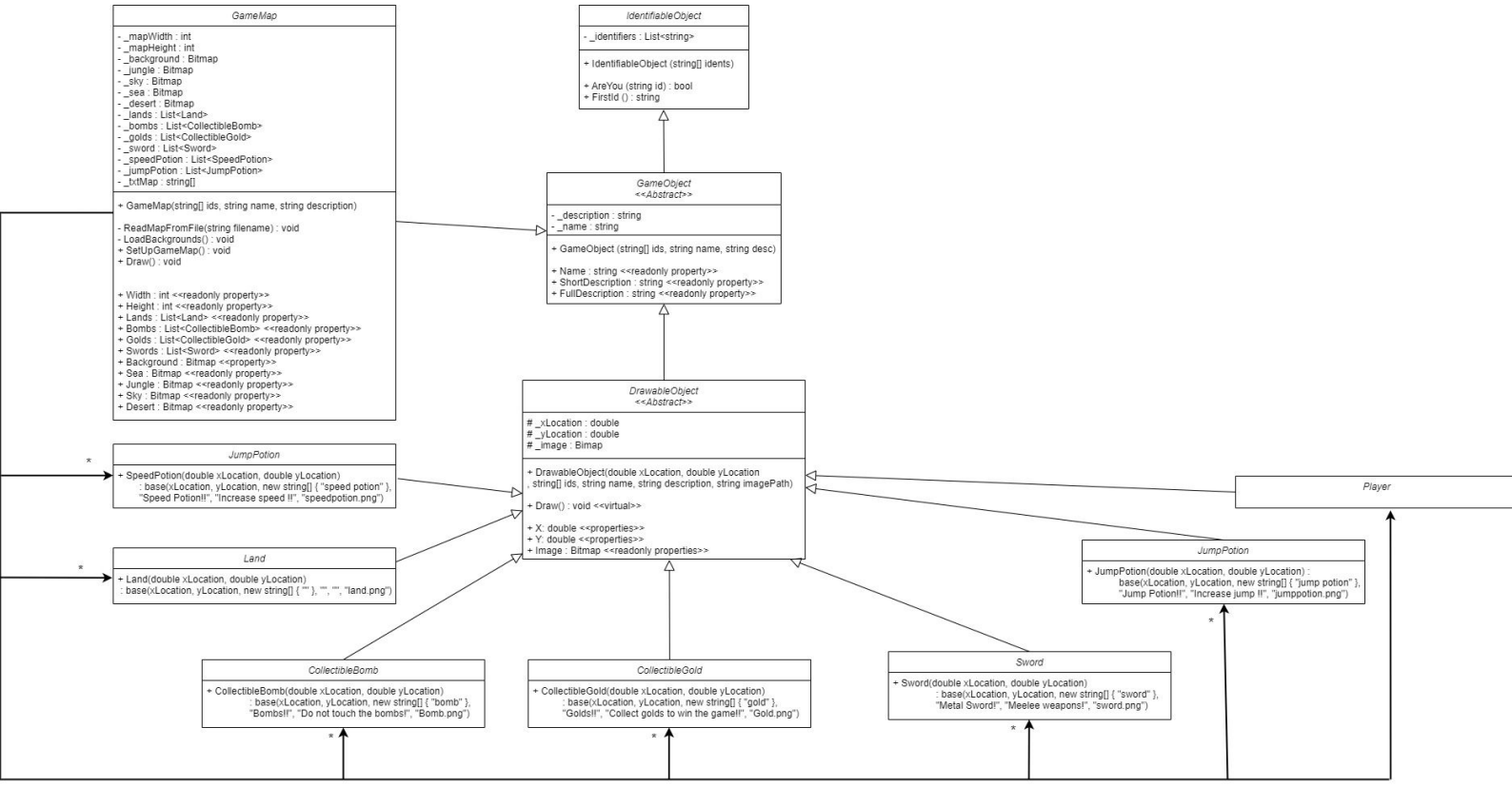
Gold.png

Bomb.png

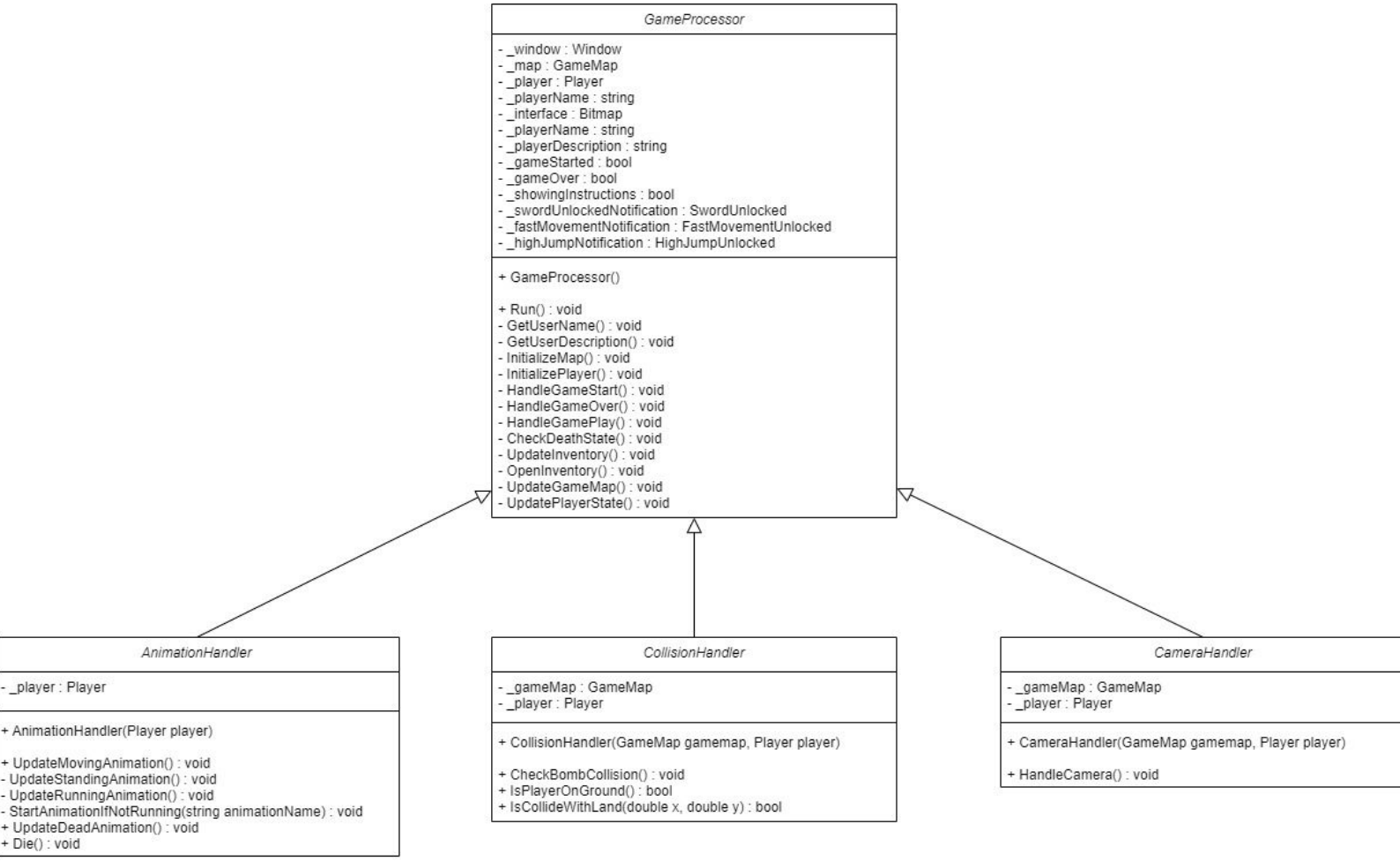
map.txt



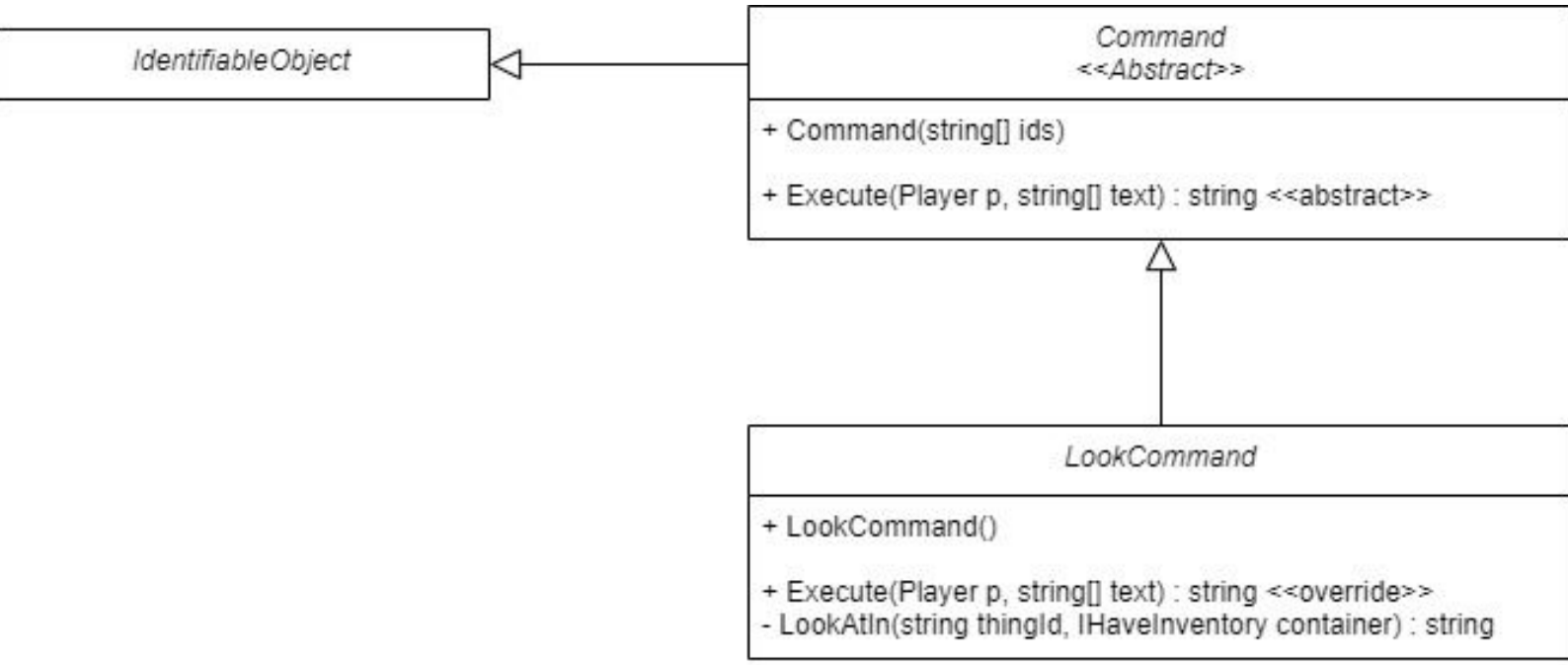
Gamemap class diagram



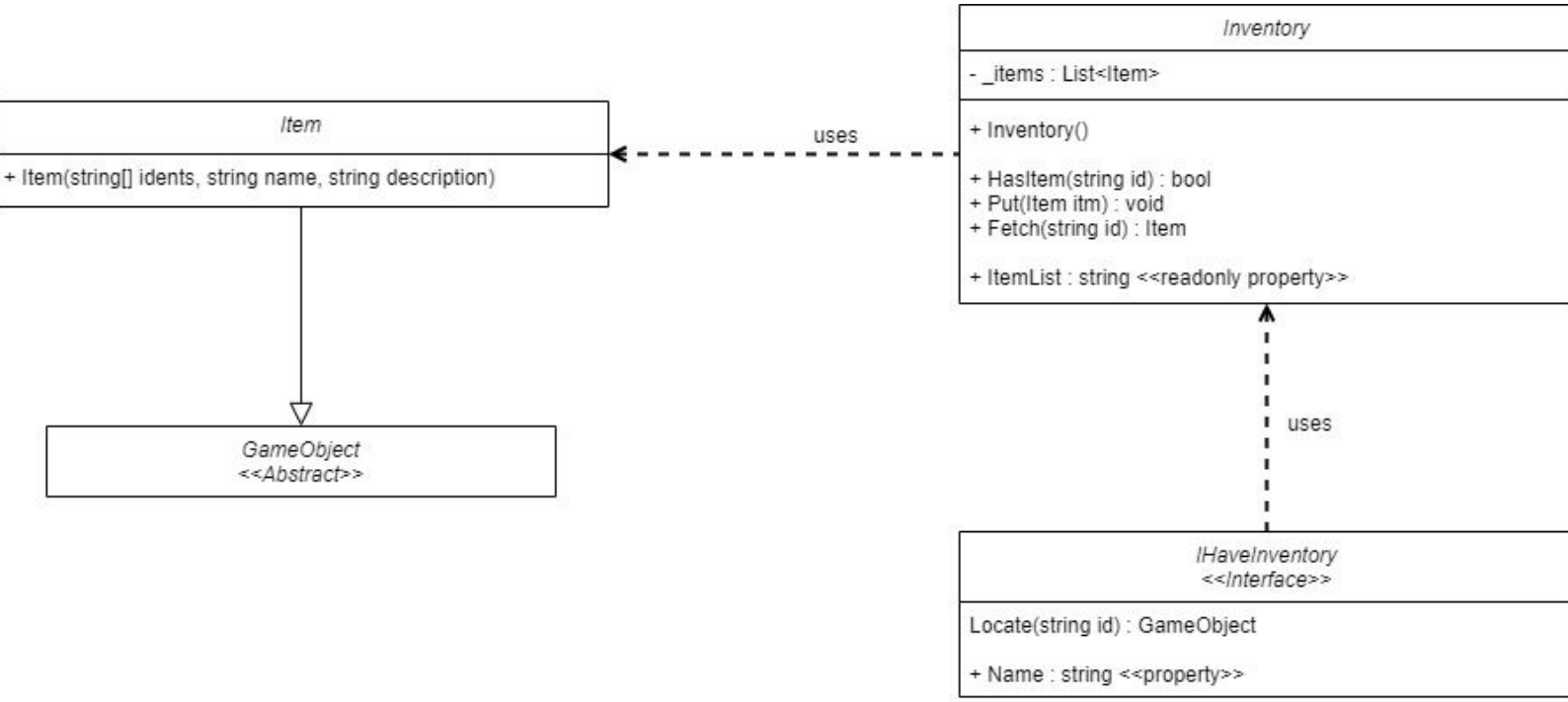
GameProcessor class diagram



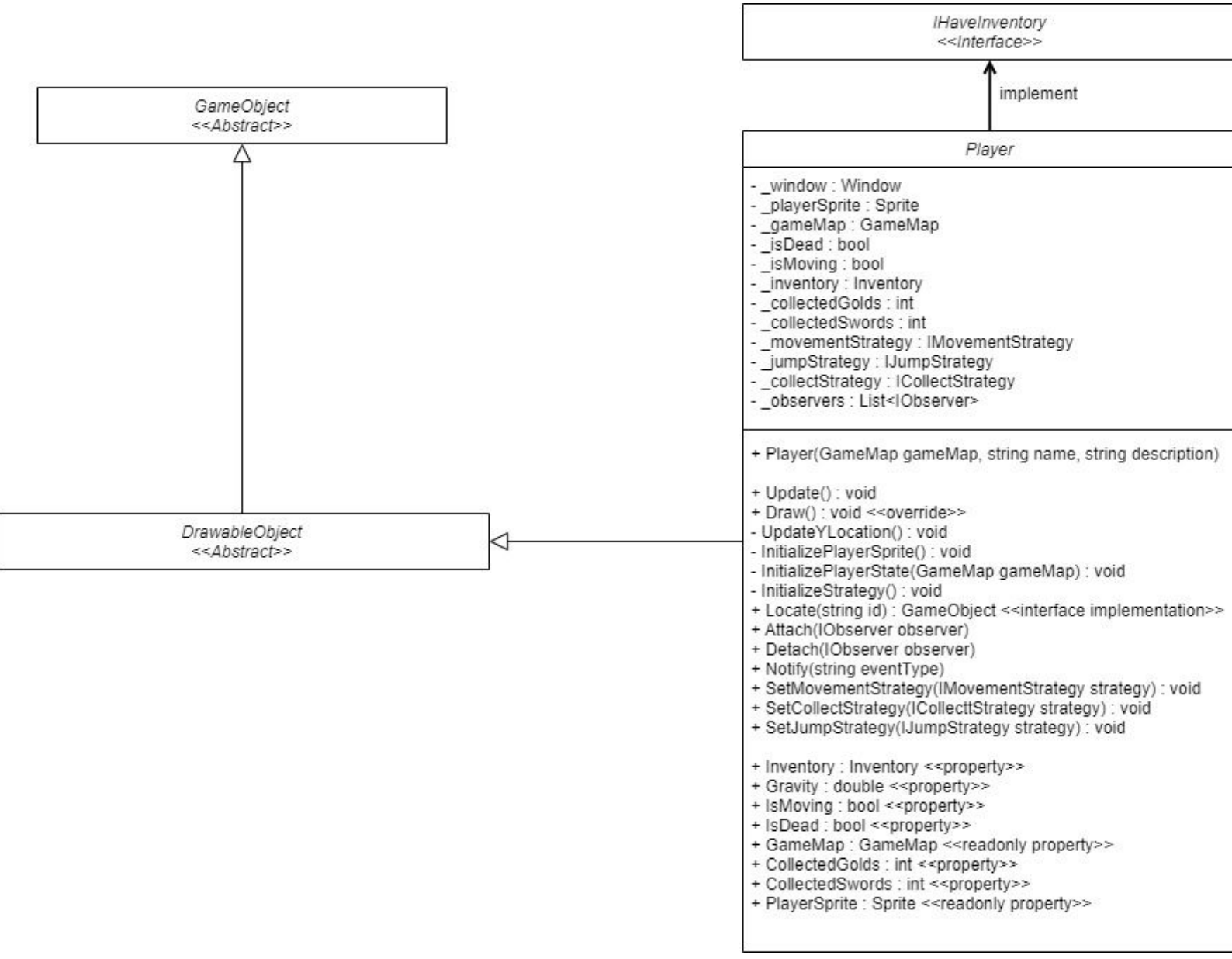
Command class diagram



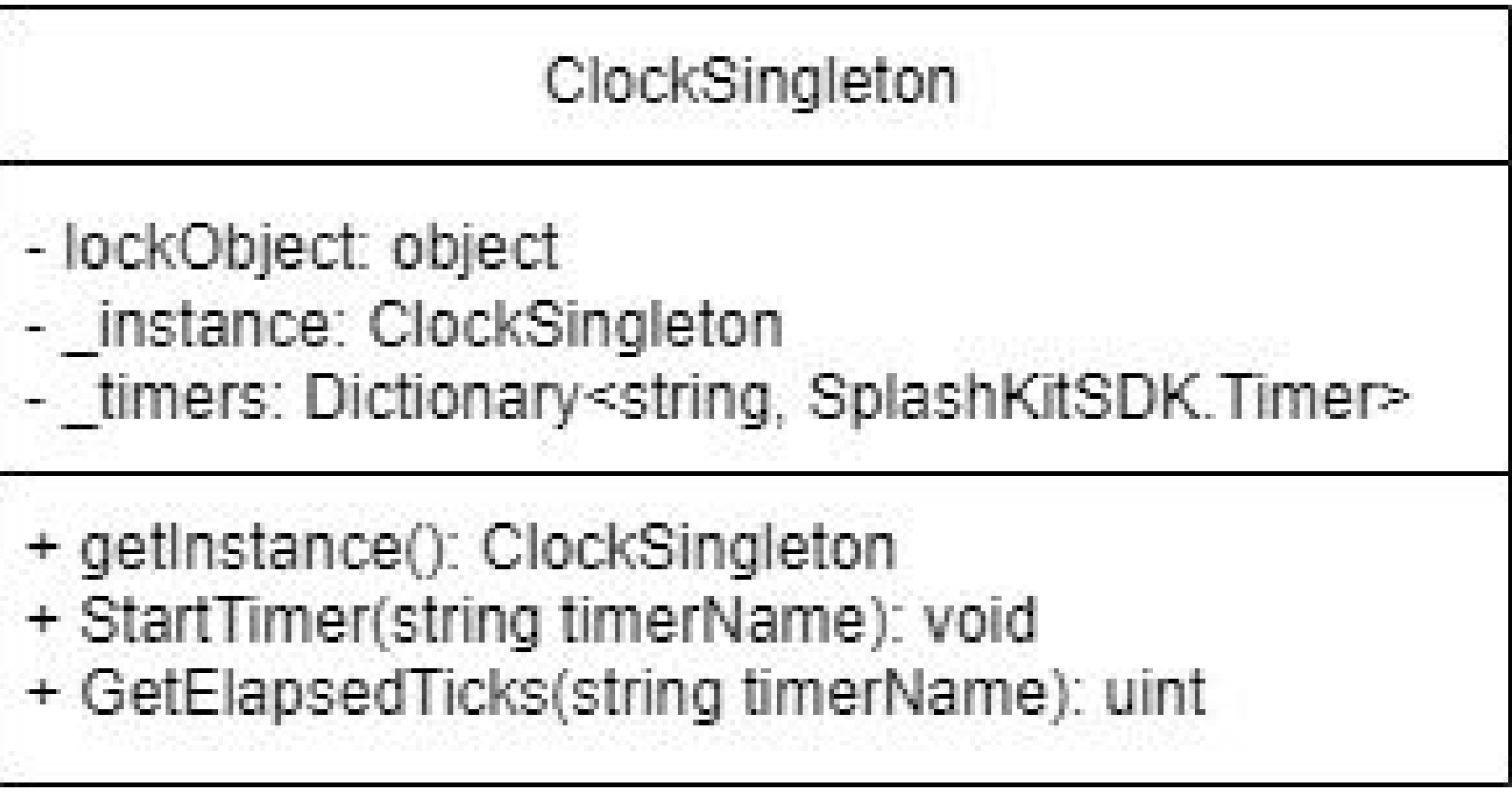
Inventory class diagram



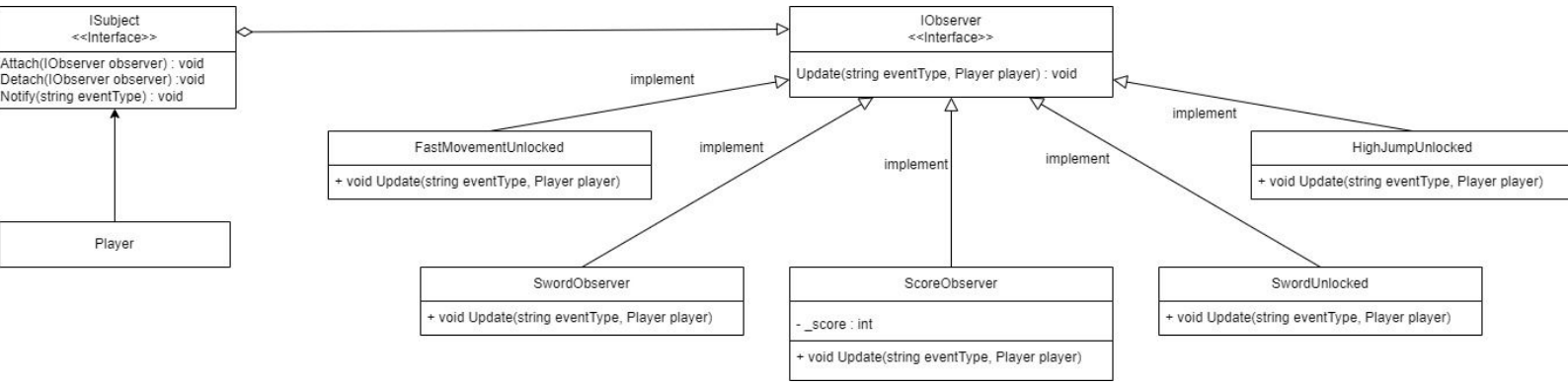
Player class diagram



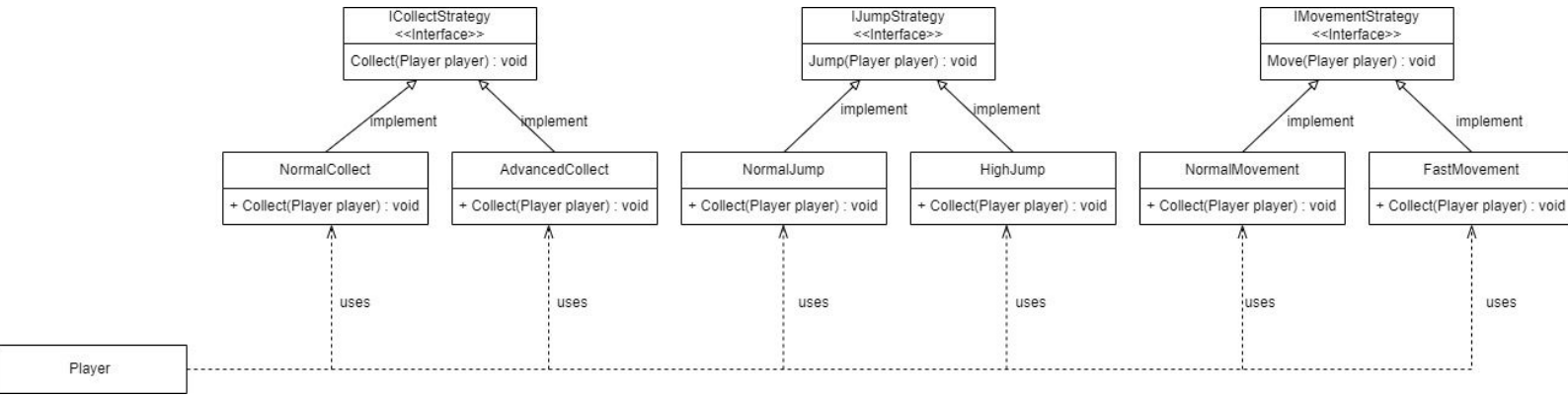
ClockSingleton class diagram



Observer design class diagram



Strategy design class diagram



Design / Usage Document

Project Overview: This project is an adventure game where the player navigates through different terrains, collects items, and interacts with the environment. The game is built using the SplashKit SDK in C#.

Design: The program uses several object-oriented programming principles and design patterns to achieve a modular and maintainable structure. The main components include:

- **GameMap Class:** Manages the game map, loading it from a text file and setting up various game objects like land, bombs, gold, swords, and potions.
- **GameProcessor Class:** Handles the game loop, initializes the game map and player, and manages game states (start, play, game over).
- **Player Class:** Represents the player character, managing its state, movement, and interactions with game objects. It implements strategies for movement, jumping, and collecting items, and uses the observer pattern to notify changes.
- **Observer Pattern:** Observers like SwordUnlocked, FastMovementUnlocked, HighJumpUnlocked, and ScoreObserver handle game events and update the game state or provide feedback to the player.
- **Strategy Pattern:** Different strategies for movement (NormalMovement, FastMovement) and jumping (NormalJump, HighJump) are implemented. The collection strategy is updated based on the player's progress.
- **Singleton Pattern:** ClockSingleton manages timers for various events.
- **Command Pattern:** Commands like LookCommand are used for player interactions with inventory.

Usage Instructions:

1. **Starting the Game:**
    - Run the Main method in Program.cs.
    - The game window will open, and you can start the game by pressing the space key.
2. **Playing the Game:**
    - Use the arrow keys to move the player.
    - Collect items like gold, swords, and potions to unlock abilities and increase your score.
    - The game map will update based on your movements and interactions.

3. **Game Over:**
    - The game will display a "Game Over" screen if the player hits a bomb.
    - Press 'R' to restart or 'Q' to quit the game.

Key Features:

- **GameMap Setup:** Reads the map layout from a text file and initializes the game objects based on the map data.
- **Player Movement:** Implements different movement strategies and handles collisions with the environment.
- **Item Collection:** Collect gold, swords, speed potions, and jump potions to enhance abilities and score.
- **Observers:** Notifies changes in the player's state, such as collecting items or unlocking abilities.
- **Game State Management:** Manages the game states and transitions between start, play, and game over screens.