```
 1  using SplashKitSDK;
 2  using System;
 3  using DragonBallGame.CharacterClass;
 4
 5  namespace DragonBallGame
 6  {
 7      public class BattleSystem
 8      {
 9          private Random _random = new Random();
10          private Character _player;
11          private Character _villain;
12          private BattleUI _battleUI;
13          private MessageManager _messageManager;
14          private VillainAI _villainAI;
15
16          public BattleSystem(Window window)
17          {
18              _battleUI = new BattleUI(window, new ImageManager());
19              _messageManager = new MessageManager();
20              _villainAI = new VillainAI();
21          }
22
23          public (string, int) BattleState(Character player, string
              difficulty)
24          {
25              _player = player;
26              _villain = GetVillainByDifficulty(difficulty);
27
28              _messageManager.AddMessage($"Battle Started: {_player.Name} vs
                  {_villain.Name} ({difficulty} Difficulty)");
29
30              while (_player.Health > 0 && _villain.Health > 0 && !
                  _battleUI.Window.CloseRequested)
31              {
32                  SplashKit.ProcessEvents();
33
34                  // Escape to Menu by pressing E
35                  if (SplashKit.KeyTyped(KeyCode.EKey))
36                  {
37                      return ($"{_player.Name} has run away!", 0);
38                  }
39
40                  SplashKit.ClearScreen(Color.White);
41
42                  _battleUI.DrawBattleState(_player, _villain);
43                  _messageManager.DrawMessages();
44
45                  ExecuteTurn();
46
```

```
47                  SplashKit.RefreshScreen(60);
48              }
49
50              string battleResult;
51              int reward = 0;
52
53              if (_villain.Health <= 0)
54              {
55                  battleResult = $"{_villain.Name} has been defeated!";
56                  reward = GetRewardByDifficulty(difficulty);
57                  _player.ResetHealth();
58                  _player.ResetEnergy();
59              }
60              else
61              {
62                  battleResult = $"{_player.Name} has been defeated...";
63                  _player.ResetHealth();
64                  _player.ResetEnergy();
65              }
66
67              return (battleResult, reward);
68          }
69
70          private Character GetVillainByDifficulty(string difficulty)
71          {
72              return difficulty switch
73              {
74                  "Easy" => CharacterFactory.CreateCharacter("Frieza"),
75                  "Medium" => CharacterFactory.CreateCharacter("Cell"),
76                  "Hard" => CharacterFactory.CreateCharacter("Buu"),
77                  "Extreme" => CharacterFactory.CreateCharacter("Black"),
78                  _ => CharacterFactory.CreateCharacter("Frieza")
79              };
80          }
81
82          private void ExecuteTurn()
83          {
84              if (_player.Health > 0)
85              {
86                  if (SplashKit.KeyTyped(KeyCode.BKey))
87                  {
88                      HandlePlayerBlock();
89                  }
90                  else if (SplashKit.KeyTyped(KeyCode.SpaceKey))
91                  {
92                      HandlePlayerAttack();
93                  }
94              }
95          }
```

```
 96
 97          private void HandlePlayerBlock()
 98          {
 99              _player.Block();
100              _messageManager.AddMessage($"{_player.Name} is blocking.");
101              _player.IncreaseEnergy(_random.Next(10, 30));
102
103              RefreshBattleUIWithDelay();
104              VillainTurn();
105          }
106
107          private void HandlePlayerAttack()
108          {
109              _player.StopBlocking();
110
111              int damage = (_player.Energy == 100) ? _player.Power / 10 :
                   _random.Next(20, _player.Power / 12);
112
113              if (_player.Energy == 100)
114              {
115                  _messageManager.AddMessage($"{_player.Name} used
                       {_player.SpecialAbility} for {damage} damage!");
116                  _player.ResetEnergy();
117              }
118              else
119              {
120                  _messageManager.AddMessage($"{_player.Name} attacks
                       {_villain.Name} for {damage} damage.");
121                  _player.IncreaseEnergy(_random.Next(10, 30));
122              }
123
124              if (_villain.IsBlocking)
125              {
126                  damage /= 2;
127                  _messageManager.AddMessage($"{_villain.Name} blocked and
                       reduced the damage by half!");
128              }
129
130              _villain.Health -= damage;
131              _villain.Health = Math.Max(_villain.Health, 0);
132
133              RefreshBattleUIWithDelay();
134              _villain.StopBlocking();
135              VillainTurn();
136          }
137
138          private void VillainTurn()
139          {
140              if (_villain.Health > 0)
```

```
141             {
142                 VillainAction action = _villainAI.Action(_villain,
                      _player);
143
144                 if (action == VillainAction.Block)
145                 {
146                     _villain.Block();
147                     _messageManager.AddMessage($"{_villain.Name} is
                       blocking.");
148                     _villain.IncreaseEnergy(_random.Next(10, 30));
149                 }
150                 else if (action == VillainAction.Attack)
151                 {
152                     HandleVillainAttack();
153                 }
154
155                 RefreshBattleUIWithDelay();
156                 _player.StopBlocking();
157             }
158         }
159
160         private void HandleVillainAttack()
161         {
162             int damage = (_villain.Energy == 100) ? _villain.Power / 10 :
                  _random.Next(20, _villain.Power / 12);
163
164             if (_villain.Energy == 100)
165             {
166                 _messageManager.AddMessage($"{_villain.Name} used
                      {_villain.SpecialAbility} for {damage} damage!");
167                 _villain.ResetEnergy();
168             }
169             else
170             {
171                 _messageManager.AddMessage($"{_villain.Name} attacks
                      {_player.Name} for {damage} damage.");
172                 _villain.IncreaseEnergy(_random.Next(10, 30));
173             }
174
175             if (_player.IsBlocking)
176             {
177                 damage /= 2;
178                 _messageManager.AddMessage($"{_player.Name} blocked and
                      reduced the damage by half!");
179             }
180
181             _player.Health -= damage;
182             _player.Health = Math.Max(_player.Health, 0);
183         }
```

```
184
185         private void RefreshBattleUIWithDelay()
186         {
187             SplashKit.ClearScreen(Color.White);
188             _battleUI.DrawBattleState(_player, _villain);
189             _messageManager.DrawMessages();
190             SplashKit.RefreshScreen(60);
191             System.Threading.Thread.Sleep(500);
192         }
193
194         private int GetRewardByDifficulty(string difficulty)
195         {
196             return difficulty switch
197             {
198                 "Easy" => 100,
199                 "Medium" => 300,
200                 "Hard" => 500,
201                 "Extreme" => 1000,
202                 _ => 0
203             };
204         }
205     }
206 }
```