```
12      public class Universe
13      {
14          #region ***** define all lists to be maintained by the Universe object *
        ****
15
16          //
17          // list of all space-time locations
18          //
19          public List<SpaceTimeLocation> SpaceTimeLocations { get; set; }
20
21          //
22          // list of all items
23          //
24          public List<Item> Items { get; set; }
25
26
27          //
28          // list of all treasure
29          //
30          public List<Treasure> Treasures { get; set; }
31
32          #endregion
33
34          #region ***** constructor *****
35
36          //
37          // default Universe constructor
38          //
39          public Universe()
40          {
41              //
42              // instantiate the lists of space-time locations and game objects
43              //
44              this.SpaceTimeLocations = new List<SpaceTimeLocation>();
45              this.Items = new List<Item>();
46              this.Treasures = new List<Treasure>();
47
48              //
49              // add all of the space-time locations and game objects to their li
         sts
50              //
51              IntializeUniverseSpaceTimeLocations();
52              IntializeUniverseItems();
53              IntializeUniverseTreasures();
54          }
55
56          #endregion
57
58          #region ***** define methods to get the next available ID for game
        elements *****
59
60          /// <summary>
61          /// return the next available ID for a SpaceTimeLocation object
62          /// </summary>
63          /// <returns>next SpaceTimeLocationObjectID </returns>
64          private int GetNextSpaceTimeLocationID()
65          {
66              int MaxID = 0;
67
68              foreach (SpaceTimeLocation STLocation in SpaceTimeLocations)
69              {
70                  if (STLocation.SpaceTimeLocationID > MaxID)
71                  {
```

```
72                      MaxID = STLocation.SpaceTimeLocationID;
73                  }
74              }
75
76              return MaxID + 1;
77          }
78
79          /// <summary>
80          /// return the next available ID for an item
81          /// </summary>
82          /// <returns>next GameObjectID </returns>
83          private int GetNextItemID()
84          {
85              int MaxID = 0;
86
87              foreach (Item item in Items)
88              {
89                  if (item.GameObjectID > MaxID)
90                  {
91                      MaxID = item.GameObjectID;
92                  }
93              }
94
95              return MaxID + 1;
96          }
97
98          /// <summary>
99          /// return the next available ID for a treasure
100         /// </summary>
101         /// <returns>next GameObjectID </returns>
102         private int GetNexTreasureID()
103         {
104             int MaxID = 0;
105
106             foreach (Treasure treasure in Treasures)
107             {
108                 if (treasure.GameObjectID > MaxID)
109                 {
110                     MaxID = treasure.GameObjectID;
111                 }
112             }
113
114             return MaxID + 1;
115         }
116
117         #endregion
118
119         #region ***** define methods to return game element objects *****
120
121         /// <summary>
122         /// get a SpaceTimeLocation object using an ID
123         /// </summary>
124         /// <param name="ID">space-time location ID</param>
125         /// <returns>requested space-time location</returns>
126         public SpaceTimeLocation GetSpaceTimeLocationByID(int ID)
127         {
128             SpaceTimeLocation spt = null;
129
130             //
131             // run through the space-time location list and grab the correct on
    e
132             //
133             foreach (SpaceTimeLocation location in SpaceTimeLocations)
```

```
            {
                if (location.SpaceTimeLocationID == ID)
                {
                    spt = location;
                }
            }

            //
            // the specified ID was not found in the universe
            // throw and exception
            //
            if (spt == null)
            {
                string feedbackMessage = $"The Space-Time Location ID {ID} doe
 s not exist in the current Universe.";
                throw new ArgumentException(ID.ToString(), feedbackMessage);
            }

            return spt;
        }

        /// <summary>
        /// get an item using an ID
        /// </summary>
        /// <param name="ID">game object ID</param>
        /// <returns>requested item object</returns>
        public Item GetItemtByID(int ID)
        {
            Item requestedItem = null;

            //
            // run through the item list and grab the correct one
            //
            foreach (Item item in Items)
            {
                if (item.GameObjectID == ID)
                {
                    requestedItem = item;
                }
            }

            //
            // the specified ID was not found in the universe
            // throw and exception
            //
            if (requestedItem == null)
            {
                string feedbackMessage = $
 "The item ID {ID} does not exist in the current Universe.";
                throw new ArgumentException(ID.ToString(), feedbackMessage);
            }

            return requestedItem;
        }

        /// <summary>
        /// get a treasure using an ID
        /// </summary>
        /// <param name="ID">game object ID</param>
        /// <returns>requested treasure object</returns>
        public Treasure GetTreasuretByID(int ID)
        {
            Treasure requestedTreasure = null;
```

```
195
196             //
197             // run through the item list and grab the correct one
198             //
199             foreach (Treasure treasure in Treasures)
200             {
201                 if (treasure.GameObjectID == ID)
202                 {
203                     requestedTreasure = treasure;
204                 };
205             }
206
207             //
208             // the specified ID was not found in the universe
209             // throw and exception
210             //
211             if (requestedTreasure == null)
212             {
213                 string feedbackMessage = $
                "The treasure ID {ID} does not exist in the current Universe.";
214                 throw new ArgumentException(ID.ToString(), feedbackMessage);
215             }
216
217             return requestedTreasure;
218         }
219
220         #endregion
221
222         #region ***** define methods to get lists of game elements by location *
        ****
223
224
225         /// get a list of items using a space-time location ID
226         /// </summary>
227         /// <param name="ID">space-time location ID</param>
228         /// <returns>list of items in the specified location</returns>
229         public List<Item> GetItemtsBySpaceTimeLocationID(int ID)
230         {
231             // TODO validate SpaceTimeLocationID
232
233             List<Item> itemsInSpaceTimeLocation = new List<Item>();
234
235             //
236             // run through the item list and put all items in the current locat
            ion
237             // into a list
238             //
239             foreach (Item item in Items)
240             {
241                 if (item.SpaceTimeLocationID == ID)
242                 {
243                     itemsInSpaceTimeLocation.Add(item);
244                 }
245             }
246
247             return itemsInSpaceTimeLocation;
248         }
249
250         /// get a list of treasures using a space-time location ID
251         /// </summary>
252         /// <param name="ID">space-time location ID</param>
253         /// <returns>list of treasures in the specified location</returns>
254         public List<Treasure> GetTreasuressBySpaceTimeLocationID(int ID)
```

```
255         {
256             // TODO validate SpaceTimeLocationID
257
258             List<Treasure> treasuresInSpaceTimeLocation = new List<Treasure>();
259
260             //
261             // run through the treasure list and put all items in the current l
    ocation
262             // into a list
263             //
264             foreach (Treasure treasure in Treasures)
265             {
266                 if (treasure.SpaceTimeLocationID == ID)
267                 {
268                     treasuresInSpaceTimeLocation.Add(treasure);
269                 }
270             }
271
272             return treasuresInSpaceTimeLocation;
273         }
274
275         #endregion
276
277         #region ***** define methods to initialize all game elements *****
278
279         /// <summary>
280         /// initialize the universe with all of the space-time locations
281         /// </summary>
282         private void IntializeUniverseSpaceTimeLocations()
283         {
284             SpaceTimeLocations.Add(new SpaceTimeLocation
285             {
286                 Name = "TARDIS Base",
287                 SpaceTimeLocationID = 1,
288                 Description = "The Norlon Corporation's secret laboratory loca
    ted deep underground, " +
289
    " beneath a nondescript 7-11 on the south-side of Toledo, OH.",
290                 Accessable = true
291             });
292
293             SpaceTimeLocations.Add(new SpaceTimeLocation
294             {
295                 Name = "Xantoria Market",
296                 SpaceTimeLocationID = 2,
297                 Description = "The Xantoria market, once controlled by the Tho
    rian elite, is now an " +
298                               "open market managed by the Xantorian Commerce C
    oop. It is a place " +
299
    "where many races from various systems trade goods.",
300                 Accessable = true
301             });
302
303             SpaceTimeLocations.Add(new SpaceTimeLocation
304             {
305                 Name = "Felandrian Plains",
306                 SpaceTimeLocationID = 3,
307                 Description =
    "The Felandrian Plains are a common destination for tourist. " +
308                   "Located just north of the equatorial line on the planet of
    Corlon, they" +
```

```
309                     "provide excellent habitat for a rich ecosystem of flora and
                    fauna.",
310                     Accessable = true
311                 });
312             }
313
314             /// <summary>
315             /// initialize the universe with all of the items
316             /// </summary>
317             private void IntializeUniverseItems()
318             {
319                 Items.Add(new Item
320                 {
321                     Name = "Key",
322                     GameObjectID = 1,
323                     Description = "A gold encrusted chest with strange markings la
                    y next to a strange blue rock.",
324                     SpaceTimeLocationID = 3,
325                     HasValue = false,
326                     Value = 0,
327                     CanAddToInventory = true
328                 });
329
330                 Items.Add(new Item
331                 {
332                     Name = "Mirror",
333                     GameObjectID = 2,
334                     Description =
                    "A full sized mirror with jewels decorating the border.",
335                     SpaceTimeLocationID = 2,
336                     HasValue = false,
337                     Value = 0,
338                     CanAddToInventory = false
339                 });
340
341                 Items.Add(new Item
342                 {
343                     Name = "Encabulator",
344                     GameObjectID = 3,
345                     Description =
                    "A multi-function device carried by all Time Lords.",
346                     SpaceTimeLocationID = 0,
347                     HasValue = true,
348                     Value = 500,
349                     CanAddToInventory = true
350                 });
351             }
352
353             /// <summary>
354             /// initialize the universe with all of the treasures
355             /// </summary>
356             private void IntializeUniverseTreasures()
357             {
358                 Treasures.Add(new Treasure
359                 {
360                     Name = "Trantorian Ruby",
361                     TreasureType = Treasure.Type.Ruby,
362                     GameObjectID = 1,
363                     Description = "A deep red ruby the size of an egg.",
364                     SpaceTimeLocationID = 2,
365                     HasValue = true,
366                     Value = 25,
367                     CanAddToInventory = true
```

```
368                 });
369
370                 Treasures.Add(new Treasure
371                 {
372                     Name = "Lodestone",
373                     TreasureType = Treasure.Type.Lodestone,
374                     GameObjectID = 2,
375                     Description = "A deep red ruby the size of an egg.",
376                     SpaceTimeLocationID = 3,
377                     HasValue = true,
378                     Value = 15,
379                     CanAddToInventory = true
380                 });
381             }
382
383             #endregion
384
385         }
```