BSc (Hons) in Information Technology
Year 2

Tutorial 6

IT2020 – Software Engineering **Semester 2, 2024**

This tutorial is designed to revise your knowledge on Design Patterns.

Activity 01 :

a) What are the three main types of design patterns?

b) Go through the simple code snippet given below. It is about a banking application of finding whether a customer is eligible to Mortgage or not.

```
Class Customer {        //get customer details

}

Class Bank {            // check whether the customer is having sufficient savings deposited in

}

Class Credit {        // check whether the customer is having a good credit limit

}

Class Loan {        // check whether the customer is having loans to be paid or any loan dues

}

 Class Mortgage { private Bank bank = new

Bank(); private Loan loan = new Loan(); private

Credit credit = new Credit();

publicboolIsEligible(Customer C)

    {
```

i) Identify the design pattern used in this code. Justify your answer.

ii) What is the main purpose of using the design pattern you have mentioned above for this solution?

Activity 02 :

"CAS" is a startup shoe manufacturing company. After operating successfully for two years, the "CAS" shoe company identified that there is a potential of expanding their production line. Therefore they decided on developing mens, sports and children shoes along with ladies shoes. They decided to provide shoes with heels, without heels, with colour designs, with plain colours and with ornaments. All kinds of possible combinations are developed and now customers are having a wide variety of selections. Some of the selections are Ladies colour designer shoes with heels,Mens flat plain colour shoes and Childrens flat colour designer shoes with ornaments.

a) Suggest a suitable design pattern which you could use for the above scenario.

b) Draw the class structure of the design pattern you identified in part (a) with appropriate methods for the above scenario.

Tutorial 6

IT2020 – Software Engineering **Semester 2, 2024**

Activity 03

"INavigator" is a GPS based road Navigation system. Read the below given description and answer the questions.

"INavigator" consists of a GPS receiver, which constantly sends a stream of $GPRMC sentences to a GPS class. A $GPRMC sentence consists of a navigation receiver warning, latitude, longitude, speed over ground, date and magnetic variation.

The GPS class continually reads, parses and stores the $GPRMC sentences as records in a buffer. These records are then displayed to the user in three views. The three views are,

- a text view which displays the basic information in the GPS sentence including distance travelled and average speed,
- a compass view which shows the direction the user is moving on and
- a breadcrumb trail view which shows the trail as well as minimum height, maximum height and the ascent (the difference between them).

a) If you were hired as a designer for this project, what would you suggest as the most appropriate design pattern to be used in implementing the system?
b) Justify your selection of the design pattern.
c) Explain your solution using a simple class diagram.

Activity 04

Find the most suitable design patterns for given scenarios.

Problem Criteria	Suitable Design Pattern
Thousands of clients are remotely invoking methods in your system. The client programs are accessing many fine-grained methods, which cause network overhead.	

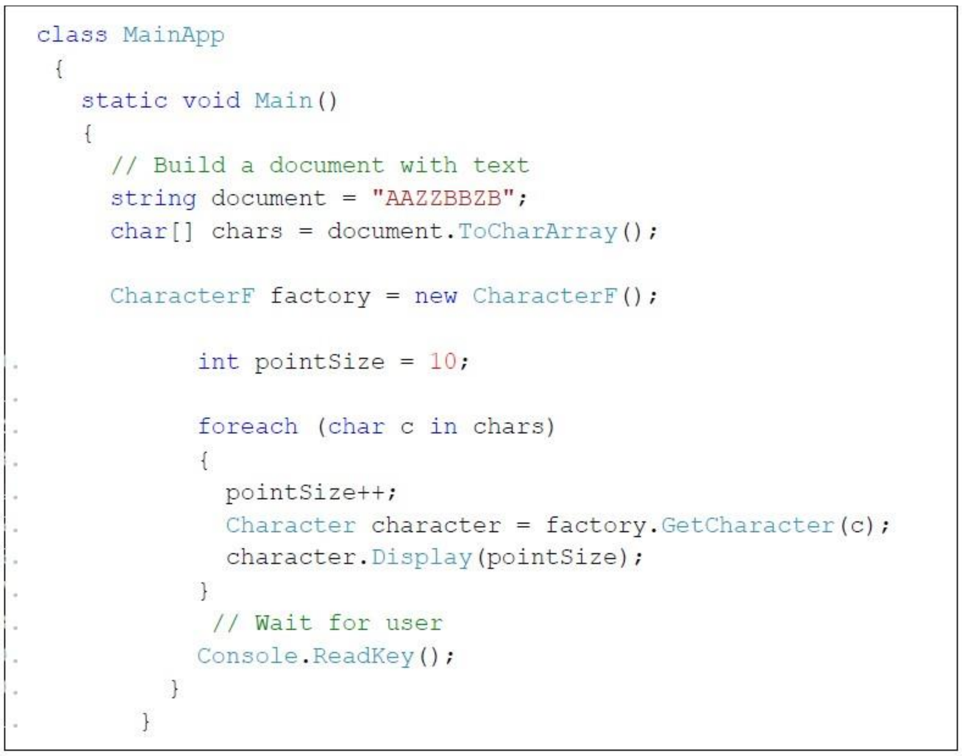
Defines one to many dependencies between objects so that when one object changes state, its entire dependents are notified and updated automatically.	
Attach additional responsibilities to an object dynamically. It provides a flexible alternative to subclassing for extending functionality.	

Self-study Questions

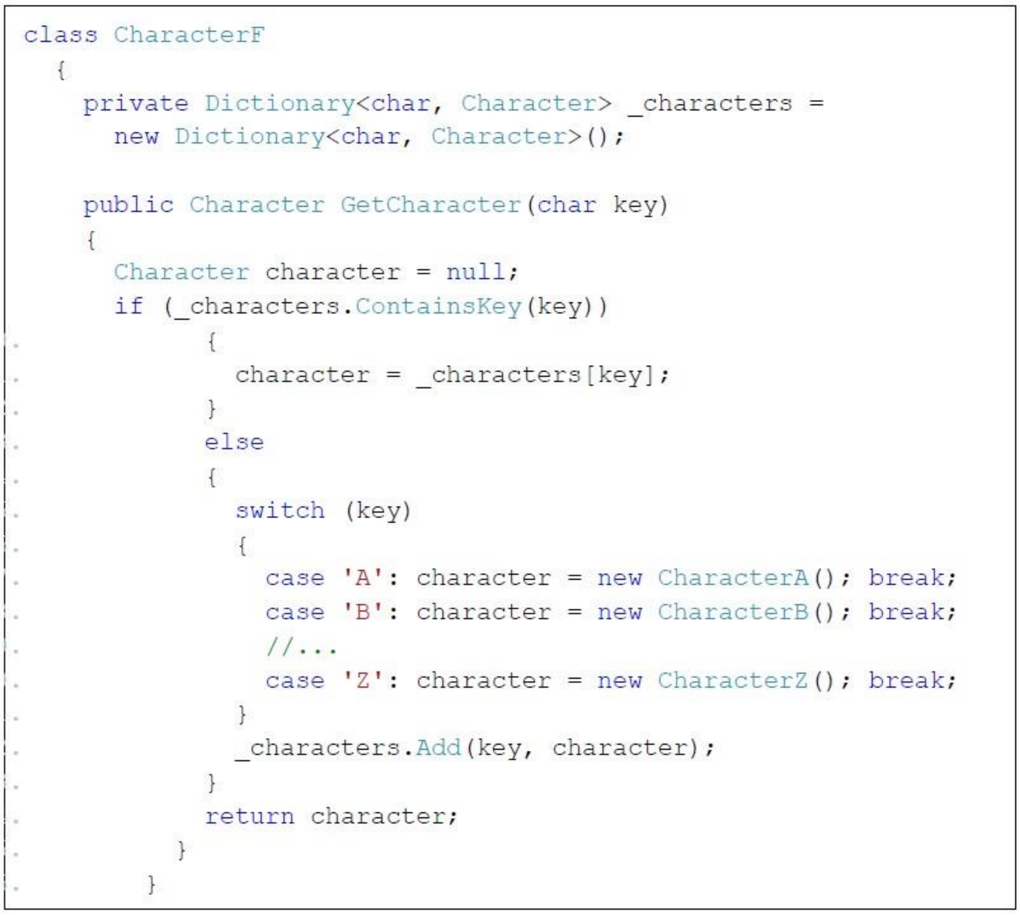
Students need to try following questions by themselves.

Activity 05

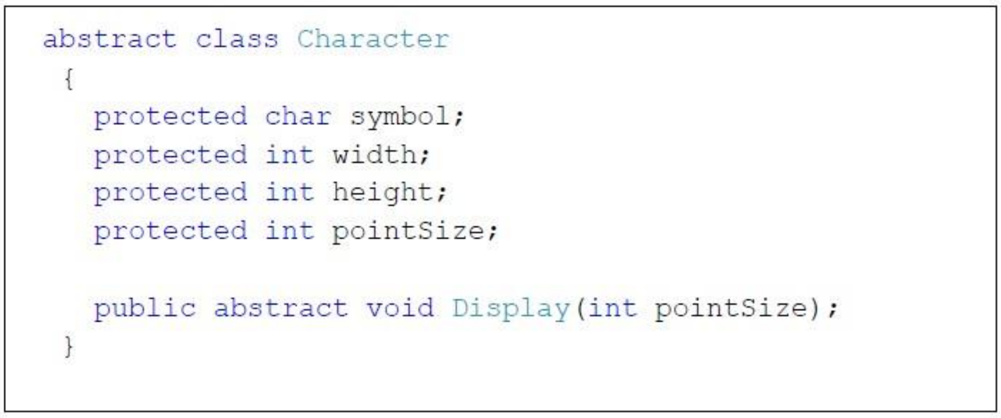
Go through the code segment given below and answer the questions.



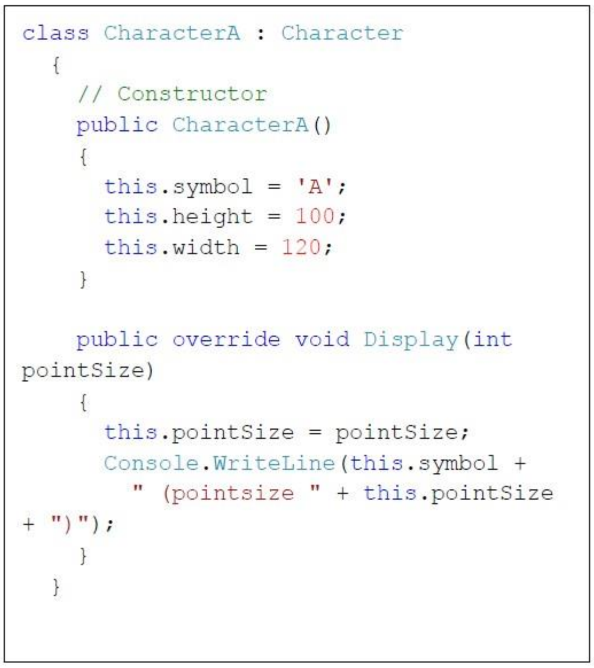
```
class MainApp
 {
   static void Main()
   {
     // Build a document with text
     string document = "AAZZBBZB";
     char[] chars = document.ToCharArray();

     CharacterF factory = new CharacterF();

           int pointSize = 10;

           foreach (char c in chars)
           {
             pointSize++;
             Character character = factory.GetCharacter(c);
             character.Display(pointSize);
           }
            // Wait for user
           Console.ReadKey();
         }
       }
```



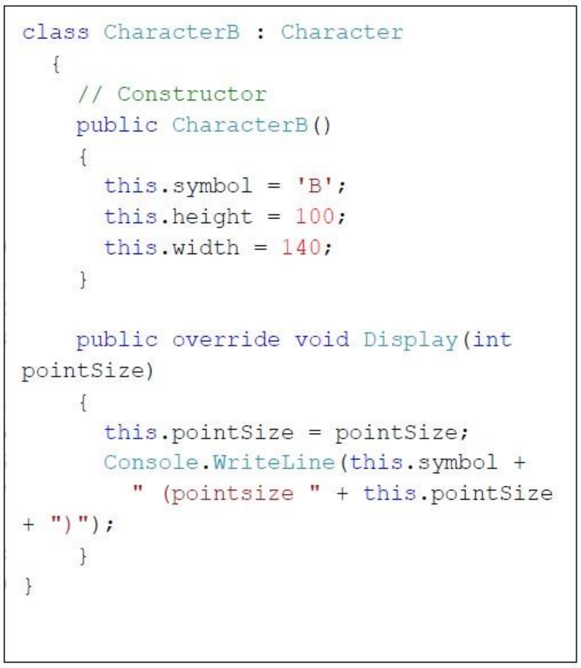
```
class CharacterF
  {
    private Dictionary<char, Character> _characters =
      new Dictionary<char, Character>();

    public Character GetCharacter(char key)
    {
      Character character = null;
      if (_characters.ContainsKey(key))
            {
              character = _characters[key];
            }
            else
            {
              switch (key)
              {
                case 'A': character = new CharacterA(); break;
                case 'B': character = new CharacterB(); break;
                //...
                case 'Z': character = new CharacterZ(); break;
              }
              _characters.Add(key, character);
            }
            return character;
          }
        }
```



```
abstract class Character
 {
   protected char symbol;
   protected int width;
   protected int height;
   protected int pointSize;

   public abstract void Display(int pointSize);
 }
```



```
class CharacterA : Character
  {
    // Constructor
    public CharacterA()
    {
      this.symbol = 'A';
      this.height = 100;
      this.width = 120;
    }

    public override void Display(int
pointSize)
    {
      this.pointSize = pointSize;
      Console.WriteLine(this.symbol +
        " (pointsize " + this.pointSize
+ ")");
    }
  }
```



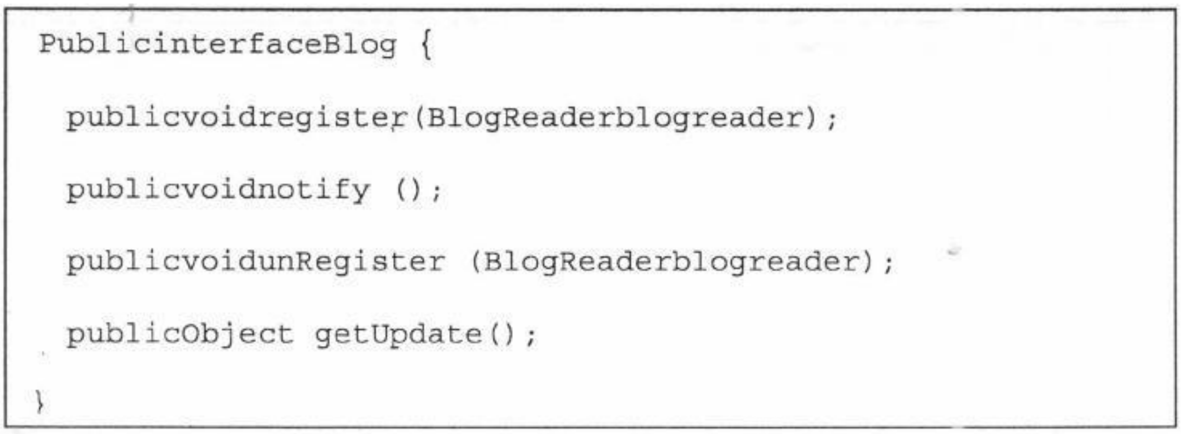
```
class CharacterB : Character
  {
    // Constructor
    public CharacterB()
    {
      this.symbol = 'B';
      this.height = 100;
      this.width = 140;
    }

    public override void Display(int
pointSize)
    {
      this.pointSize = pointSize;
      Console.WriteLine(this.symbol +
        " (pointsize " + this.pointSize
+ ")");
    }
}
```

a) What is the design pattern that has been used to implement this code segment?

b) Using a simple class diagram, show the structure of the design pattern you defined in part a) above. Indicate the main methods given in the code segment on your class diagram.

Activity 06

Go through the code segment given below and answer the questions.



```
PublicinterfaceBlog {

  publicvoidregister(BlogReaderblogreader);

  publicvoidnotify ();

  publicvoidunRegister (BlogReaderblogreader);

  publicObject getUpdate();

}
```

BSc (Hons) in Information Technology
Year 2

Tutorial 6

IT2020 – Software Engineering **Semester 2, 2024**

```
importjava.util.ArrayList;
importjava.util.List;

publicclassBlog implementsRealBlog {

  List<> BlogList;
  privatebooleanstateChange;

  publicBlog() {
    this.BlogList = newArrayList<>();
    stateChange = false;
  }

  publicvoidregister (BlogReaderblogreader) {
    BlogList.add(blogreader);
  }

  publicvoidunRegister (BlogReaderblogreader) {
    BlogList.remove(blogreader);
  }

  publicvoidnotify() {

    if(stateChange) {
      for(BlogReaderblogreader: BlogList) {
        blogreader.update();
      }
    }
  }

  publicObject getUpdate() {
    Object changedState = null;
    if(stateChange) {
      changedState = "XXX Design Pattern";
    }
    returnchangedState;
  }

  publicvoidpostNewArticle() {
    stateChange = true;
    notify();
  }
}
```

SLIIT
Discover Your Future

BSc (Hons) in Information Technology
Year 2

Tutorial 6

IT2020 – Software Engineering **Semester 2, 2024**

```
publicinterfaceBlogReader {

  publicvoidupdate();

  publicvoidsetSubject(Subject subject);
}
```

```
publicclassUser implementsBlogReader {

  privateString article;
  privateSubject blog;

  publicvoidsetSubject(Subject blog) {
    this.blog = blog;
    article = "No New Article!";
  }

  @Override
  publicvoidupdate() {
    System.out.println("State change reported by Subject.");
    article = (String) blog.getUpdate();
  }

  publicString getArticle() {
    returnarticle;
  }
}
```

a) What is the design pattern that has been used to implement the above code segment?

b) Using a simple class diagram, show the structure of the design pattern you defined in part (a) above.