**EXERCISE 2:**I mplementing the Factory Method Pattern

**Program:**

**INotification.cs:**

An interface that likely defines a method like Send() which all notification types must implement.

**Code:**

public interface INotification

{

void Send(string message);

}

|  |
| --- |
| **INotificationCreator.cs** |

|  |
| --- |
| Interface for notification creator classes (abstract factory), which define a method like CreateNotification(). |

**Code:**

public abstract class NotificationCreator

{

public abstract INotification CreateNotification();

public void Notify(string message)

{

INotification notification = CreateNotification();

notification.Send(message);

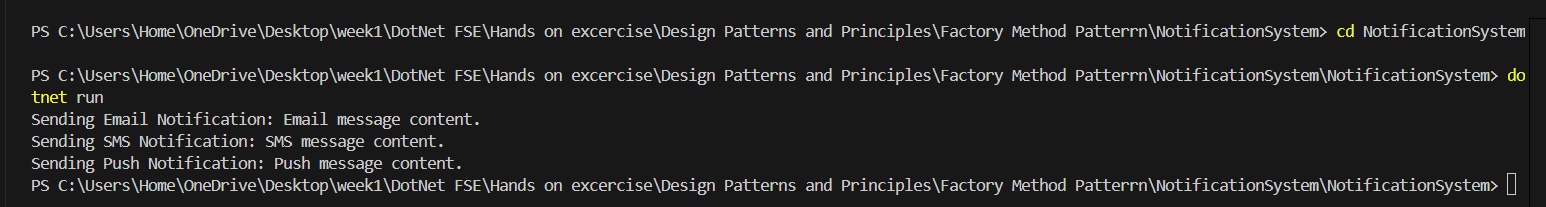
}

}

|  |
| --- |
| **EmailNotification.cs** |
|  |

|  |
| --- |
| A class that implements INotification and handles logic for sending email notifications. |
|  |
|  |
| **Code:**  public class EmailNotification : INotification  {  public void Send(string message)  {  Console.WriteLine("Sending Email Notification: " + message);  }  }   |  | | --- | | **EmailNotificationCreator.cs** | |  | |  |  |  | | --- | | A factory class that creates instances of EmailNotification. | | **Code:**  public class EmailNotificationCreator : NotificationCreator  {  public override INotification CreateNotification()  {  return new EmailNotification();  }  }   |  | | --- | | **SmsNotification.cs** | |  |  |  |  |  | | --- | --- | --- | | Represents the SMS notification logic, probably implements INotification.  **Code:**  public class SmsNotification : INotification  {  public void Send(string message)  {  Console.WriteLine("Sending SMS Notification: " + message);  }  }   |  | | --- | | **SmsNotificationCreator.cs** |  |  | | --- | | Creates SmsNotification objects, following the factory method pattern.  **Code:** | |   public class SmsNotificationCreator : NotificationCreator  {  public override INotification CreateNotification()  {  return new SmsNotification();  }  }   |  | | --- | | **PushNotification.cs** |  |  | | --- | | Implements push notification logic (e.g., app alerts). | | **Code:**  public class PushNotification : INotification  {  public void Send(string message)  {  Console.WriteLine("Sending Push Notification: " + message);  }  }   |  | | --- | | **PushNotificationCreator.cs** |  |  | | --- | | Factory class for creating push notifications. | | **Code:** | |   public class PushNotificationCreator : NotificationCreator  {  public override INotification CreateNotification()  {  return new PushNotification();  }  }   |  | | --- | | **NotificationCreator.cs** |  |  |  |  | | --- | --- | --- | | An abstract or base class implementing INotificationCreator that other creators inherit from.  **Code:**  public abstract class NotificationCreator  {  public abstract INotification CreateNotification();  public void Notify(string message)  {  INotification notification = CreateNotification();  notification.Send(message);  }  }   |  |  | | --- | --- | | **Program.cs**  The main entry point of the application, where different types of notifications are created and tested based on user input or configuration.  **Code:**  using System;  class Program  {  static void Main(string[] args)  {  NotificationCreator emailCreator = new EmailNotificationCreator();  emailCreator.Notify("Email message content.");  NotificationCreator smsCreator = new SmsNotificationCreator();  smsCreator.Notify("SMS message content.");  NotificationCreator pushCreator = new PushNotificationCreator();  pushCreator.Notify("Push message content.");  }  } |  | | | |  | |
|  |

**Output:**

****