1. (Calculate Metrics of Circle and Cylinder)(30 Points)

Given the definition of the abstract class **Shape**, You are supposed to implement its subclasses **Circle** and **Cylinder**. The base class **Shape** is defined as below.

```
public abstract class Shape {
    // fields
    protected String name;

    // methods
    public Shape(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    // abstract method
    public abstract double getArea();
    public abstract void showDescription();
}
```

- fields:
  - name : **String**, the Shape's name
- Methods :
  - getArea() : return the area of the Shape
  - showDescription(): Output the details **to the console**, and all the non-integer should be rounded up to **4 decimal places**.

The detail of subclass **Circle** and the subclass **Cylinder** is listed as below.

Class Circle extends Shape:

- Fields :
  - name : **String**, inherited from Base Class
  - radius : **double**, the radius of the Circle
- Methods :
  - Circle(double radius) : the constructor of Circle
  - getRadius() : return the radius of the Circle
  - setRadius(double radius) : reset the radius of the Circle
  - getArea() : abstract method that **inherited** from the Base Class and **shall be implemented**
  - getPerimeter() : return the perimeter of the Circle
  - showDescription() : abstract method that **inherited** from the Base Class and **shall be implemented**

Class Cylinder extends Shape:

- Fields :
  - name : **String**, inherited from Base Class
  - radius : **double**, the radius of the Cylinder
  - height : **double**, the height of the Cylinder
- Methods :
  - Cylinder(double radius, double height) : the constructor of Cylinder
  - getHeight() : return the height of the Cylinder
  - setHeight(double height) : reset the height of the Cylinder
  - getRadius() : return the radius of the Cylinder
  - setRadius(double radius) : reset the radius of the Cylinder
  - getArea() : abstract method that **inherited** from the Base Class and **shall be implemented**
  - getVolume() : return the volume of the Cylinder
  - showDescription() : abstract method that **inherited** from the Base Class and **shall be implemented**

Requirements:

- You should **implement the class** and **complete the test program** as shown below.
- The Classes must contain methods and fields mentioned above, and it also works if your implement has more methods and fields than the requirement.

ShapeTest.java:

- test code :

```
public class ShapeTest {
    public static void main(String args[]){
        Shape shape1 = new Circle(3.0);
        Shape shape2 = new Cylinder(3.0, 4.0);

        double sumAreaOfShape = ________________;
        System.out.println("Sum area of shape is: " + ________________);
        System.out.println("The name of shape1 is: " + ________________);
        System.out.println("The name of shape2 is: " + ________________);
        System.out.println("");

        shape1.________________;
        shape2.________________;
    }
}
```

- output :

```
Sum area of shape is: 160.2212
The name of shape1 is: Circle
The name of shape2 is: Cylinder

Shape: Circle
radius: 3.0000
```

```
Area: 28.2743
Perimeter: 18.8496

Shape: Cylinder
radius: 3.0000
height: 4.0000
Area: 131.9469
Volume: 113.0973
```

2. (The Sender And Receiver)(30 Points)

In social media software, users could send messages to each other. They could also send messages to a channel, and then all users on that channel would receive those messages. Implement this situation. It contains two interfaces and two classes as below.

Interface Receiver:

- Methods :
  - receive(Sender sender, String message)

Interface Sender:

- Methods :
  - send(Receiver receiver, String message) :

Class User implements Sender, Receiver:

- Fields :
  - name : **String**, the name of user
  - messageList : **List<String>**, the messages that user received
- Methods :
  - send(Receiver receiver, String message) : method that **inherited** from the Interface and **shall be implemented,** send message to receiver in the form of "**[From <User's name>]<message>**"
  - receive(Sender sender, String message) : method that **inherited** from the Interface and **shall be implemented,** append message to messageList.
  - showMessages() : print messages in messageList line by line **to console**.

Class Channel implements Receiver:

- Fields :
  - name : **String**, the channel's name
  - userList : **List<User>**, the users in this channel
- Methods :
  - receive(Sender sender, String message) : method that **inherited** from the Interface and **shall be implemented,** transfer message to all users except the sender in this channel in the form of "**[Channel <Channel's name>]<message>**"
  - add(User user) : append user to userList
  - remove(User user) : remove user from userList

Requirements:

- You should **implement the class** and **complete the test program** as shown below.
- The Classes must contain methods and fields mentioned above, and it also works if your implement has more methods and fields than the requirement.

Test program:

- test code :

```
public class SenderAndReceiverTest {
    public static void main(String[] args) {
        User zhangSan = new User("ZhangSan");
        User liSi = new User("LiSi");
        User wangWu = new User("WangWu");
        User zhaoLiu = new User("zhaoLiu");
        Channel douBiFanZhiJiDi = new Channel("douBiFanZhiJiDi");
        ________.add(________);
        ________.add(________);
        ________.add(________);
        Channel yongYuanDe308 = new Channel("yongYuanDe308");
        ________.add(________);
        ________.add(________);
        ________.add(________);

        ________.send(________, "What did you guys do during the holidays");
        ________.send(________, "I went to eat barbecue, haha");
        ________.send(________, "It sounds interesting, we can barbecue
together next time!");
        ________.remove(________);

        ________.send(________, "Please help me ask what Zhao Liu likes");
        ________.send(________, "Zhang, I was Wang. Can't help you, bro.
sorry");
        ________.send(________, "Lisi, would you mind...");
        ________.send(________, "You know I don’t like to be a
matchmaker.");

        System.out.println("ZhangSan's messages list:");
        ________.________();
        System.out.println("\nLiSi's messages list:");
        ________.________();
        System.out.println("\nWangWu's messages list:");
        ________.________();
        System.out.println("\nZhaoLiu's messages list:");
        ________.________();
    }
}
```

- output :

```
ZhangSan's messages list:
[From douBiFanZhiJiDi][From WangWu]Zhang, I was Wang. Can't help you, bro. sorry
[From douBiFanZhiJiDi][From LiSi]You know I don’t like to be a matchmaker.

LiSi's messages list:
[From yongYuanDe308][From zhaoLiu]What did you guys do during the holidays
[From yongYuanDe308][From WangWu]I went to eat barbecue, haha
[From yongYuanDe308][From zhaoLiu]It sounds interesting, we can barbecue together next time!
[From douBiFanZhiJiDi][From ZhangSan]Please help me ask what Zhao Liu likes
[From douBiFanZhiJiDi][From WangWu]Zhang, I was Wang. Can't help you, bro. sorry
[From douBiFanZhiJiDi][From ZhangSan]Lisi, would you mind...

WangWu's messages list:
[From yongYuanDe308][From zhaoLiu]What did you guys do during the holidays
[From yongYuanDe308][From zhaoLiu]It sounds interesting, we can barbecue together next time!
[From douBiFanZhiJiDi][From ZhangSan]Please help me ask what Zhao Liu likes
[From douBiFanZhiJiDi][From ZhangSan]Lisi, would you mind...
[From douBiFanZhiJiDi][From LiSi]You know I don’t like to be a matchmaker

ZhaoLiu's messages list:
[From yongYuanDe308][From WangWu]I went to eat barbecue, haha
```

3. (The Observer Pattern)(40 Points)

In WeiBo, if user A clicks the **Follow Button** at user B’s home page, then three events will happen:

1. user B will appear in user A’s follow list (**FollowIncrease**).
2. user A will appear in user B’s fans list (**FansIncrease**).
3. user B will be notified that user A has followed him/her (**BeFollowed**).

Implement this situation using **Observer Pattern**. It contains five classes and one interface as below.

Class User:

- Fields :
  - name : **String**, the name of user
  - followList : **List<User>**, the follow list of user
  - fansList : **List<User>**, the fans list of user

- messageList : **List<String>**, the messages list of user

- Methods :
  - User(String name) : the constructor
  - getName() : return user's name
  - getFollowList() : return user's follow list
  - getFansList() : return user's fans list
  - notify(String message) : append message to messages list
  - showFollowList() : output each user's name in follow list line by line **to console**
  - showFansList() : output each user's name in fans list line by line **to console**
  - ShowMessageList() : output each message in messages list line by line **to console**

Class FollowButton:

- Fields :
  - pageUser : **User**, the follow button is at this user's home page
  - observerList : **List<Observer>**, the observer list
- Methods :
  - FollowButton(User pageUser) : the constructor
  - click(User clicker) : click this button by clicker
  - addObserver(Observer observer) : add observer to observerList

Interface Observer:

- Methods :
  - notify(User pageUser, User follower)

Class FollowIncrease implements Observer:

- Methods :
  - notify(User pageUser, User follower) : method that **inherited** from the Interface and **shall be implemented, which will** increase the follower's follow list

Class FansIncrease implements Observer:

- Methods :
  - notify(User pageUser, User follower) : method that **inherited** from the Interface and **shall be implemented, which will** increase the pageUser's fans list

Class BeFollowed implements Observer:

- Methods :
  - notify(User pageUser, User follower) : method that **inherited** from the Interface and **shall be implemented, which will** notify the pageUser with a message in the form of "**You are followed by <follower's name>**"

Requirements:

- You should **implement the class** and **complete the test program** as shown below.
- The Classes must contain methods and fields mentioned above, and it also works if your implement has more methods and fields than the requirement.

Test program:

- test code :

```
public class ObserverPatternTest {
```

```
    public static void main(String[] args) {
        User zhangSan = new User("ZhangSan");
        User liSi = new User("LiSi");
        User wangWu = new User("WangWu");
        User zhaoLiu = new User("zhaoliu");
        FollowButton[] followButtons = ________;
        FollowIncrease followIncrease = new FollowIncrease();
        FansIncrease fansIncrease = new FansIncrease();
        BeFollowed beFollowed = new BeFollowed();
        for (FollowButton followButton : followButtons) {
            ________;
            ________;
            ________;
        }
        followButtons[0].click(________);
        followButtons[0].click(________);
        followButtons[0].click(________);
        followButtons[1].click(________);
        followButtons[1].click(________);
        followButtons[2].click(________);
        followButtons[2].click(________);
        followButtons[2].click(________);
        followButtons[3].click(________);
        followButtons[3].click(________);

        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
        ________.________ ();
    }
}
```

- output :

```
ZhangSan's fans list:
LiSi
```

```
WangWu
zhaoliu
ZhangSan's follow list:
LiSi
WangWu
ZhangSan's message list:
You are followed by LiSi
You are followed by WangWu
You are followed by zhaoliu
LiSi's fans list:
ZhangSan
WangWu
LiSi's follow list:
ZhangSan
WangWu
zhaoliu
LiSi's message list:
You are followed by ZhangSan
You are followed by WangWu
WangWu's fans list:
ZhangSan
LiSi
zhaoliu
WangWu's follow list:
ZhangSan
LiSi
zhaoliu
WangWu's message list:
You are followed by ZhangSan
You are followed by LiSi
You are followed by zhaoliu
zhaoliu's fans list:
LiSi
WangWu
zhaoliu's follow list:
ZhangSan
WangWu
zhaoliu's message list:
You are followed by LiSi
You are followed by WangWu
```