1- Which of the following is true?

(a) A finally block is executed before the catch block but after the try block.
(b) A finally block is executed, only after the catch block is executed.
(c) A finally block is executed whether an exception is thrown or not.
(d) A finally block is executed, only if an exception occurs.
(e) None of the above.

2- Consider the following code

```
Rectangle r1 = new Rectangle();
r1.setColor(Color.blue);
Rectangle r2 = r1;
r2.setColor(Color.red);
```

After the above piece of code is executed, what are the colors of r1 and r2 (in the same order)?

a) blue and red
b) blue and blue
c) red and red
d) red and blue
e) none of the above

3- which statement is NOT true?

(a) A class containing abstract methods is called an abstract class.
(b) Abstract methods should be implemented in the derived class.
(c) An abstract class cannot have non-abstract methods.
(d) A class must be qualified as 'abstract' class, if it contains one abstract method.
(e) None of the above.

4- The fields in an interface are implicitly specified as:

a. static only
b. protected
c. private
d. both static and final
e. none of the above

5- you are going to implement a system for sending 2 types of messages (Email and SMS) This system checks if the receiver and sender information are valid and then it sends the message either by email or by sms.

Follow these steps to create the required classes and interface.

Create class **Address**:

```
private String streetAddress;
private int bulidingNumber;
```

Add required getters, setters and constructor and methods.

Create Class **User** with following fields:

```
private String firstName;
private String lastName;
private Address address;
```

Add required Constructors, Getters and Setters and other required methods.
It has 2 children (SmsUser and EmailUser)
EmailUser has one field (emailAddress string).
SmsUser has one field (phoneNumber string).
Add Validation in the setPhoneNumber to check if the phone number only contains digits otherwise, throws IllegalArgumentException.

Create a class **Message**

```
private User reciever;
private User sender;
private String body;
```

Add required Constructors, Getters and Setters and other required methods.
This class has 2 children EmailMessage and SmsMessage. These classes don't have any fields and they only extend class Message.

Create an interface name it as ISendInfo which has only on

```
public interface ISendInfo {
    boolean validateMessage(User sender, User receiver, String body);

    void sendMesage(Message message);
}
```

Create 2 implementation of this interface which has different **implementation** for the IsendInfo.

** Read carefully the comments in the body of the methods, it shows what you need to do for each method!!!

SendEmail:

```java
@Override
public boolean validateMessage(User sender, User receiver, String body) {
    //validate if the email addresses are correct
    //it needs to contain (@ and .) only one time in the email address
    //if you use regex would be a bonus ;)

    //also

    //check if the message body is not empty
    //check if it does not contain (^ or * or !) in the message body
    //and if it contains throw IllegalArgumentException

    return true;
}

@Override
public void sendMessage(Message message) {
    //save the information of the message
    //which is the message body and sender and receiver
    //into a file name it as 'email.txt'
}
```

SendSms:

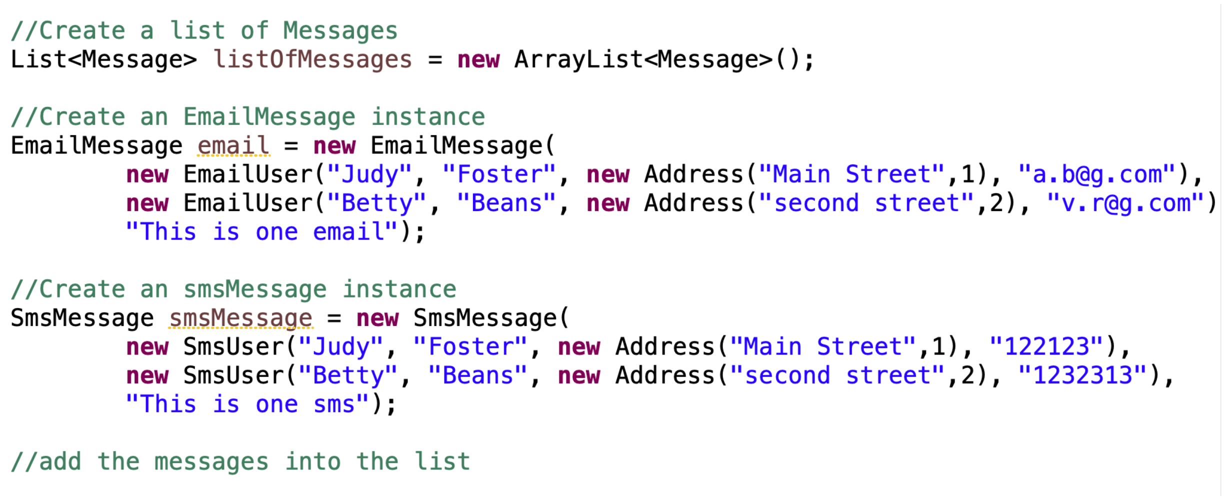
```java
@Override
public boolean validateMessage(User sender, User receiver, String body) {
    //implement the body to validate if the user has
    //a correct phone number 10 digit
    // Bonus if you use regex for the phone number ;)
    //and if it is not validate then throw IllegalArgumentException
    //with a proper message

    //also

    //check if the body is not empty
    // if it is not longer than 160 characters
    //if it does not contain any (& or # or @) in the message
    //and if it is not validate throw IllegalArgumentException
    //with a proper message
    return true;
}

@Override
public void sendMesage(Message message) {
    //save the information of the message
    //which is the message body and sender and receiver
    //information into a file name it as 'sms.txt'
}
```

Finally, create a list of messages and add them to a list.
For each member of the list, iterate through the list and call **sendMesage** if **validateMessage** result return true, otherwise print a proper message to show that the message is invalid!



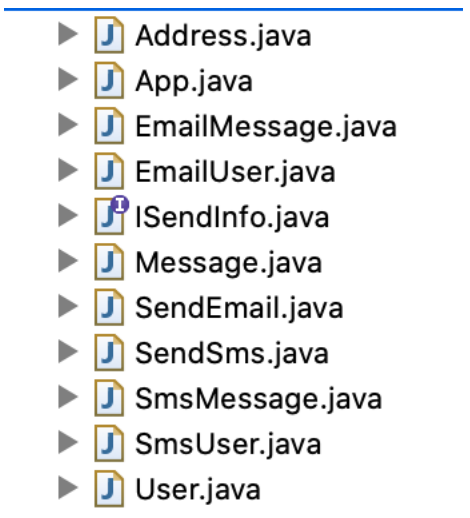
```
//Create a list of Messages
List<Message> listOfMessages = new ArrayList<Message>();

//Create an EmailMessage instance
EmailMessage email = new EmailMessage(
        new EmailUser("Judy", "Foster", new Address("Main Street",1), "a.b@g.com"),
        new EmailUser("Betty", "Beans", new Address("second street",2), "v.r@g.com")
        "This is one email");

//Create an smsMessage instance
SmsMessage smsMessage = new SmsMessage(
        new SmsUser("Judy", "Foster", new Address("Main Street",1), "122123"),
        new SmsUser("Betty", "Beans", new Address("second street",2), "1232313"),
        "This is one sms");

//add the messages into the list
```

At the end, I expect to see 10 classes and 1 interface in your project ;)



```
▶ Address.java
▶ App.java
▶ EmailMessage.java
▶ EmailUser.java
▶ ISendInfo.java
▶ Message.java
▶ SendEmail.java
▶ SendSms.java
▶ SmsMessage.java
▶ SmsUser.java
▶ User.java
```

Good luck!