Homework 4-1: Create an object-oriented validation class

Console

```
Welcome to the Validation Tester application

Int Test
Enter an integer between -100 and 100: x
Error! Invalid integer value. Try again.
Enter an integer between -100 and 100: -101
Error! Number must be greater than -101
Enter an integer between -100 and 100: 101
Error! Number must be less than 101
Enter an integer between -100 and 100: 100

Double Test
Enter any number between -100 and 100: x
Error! Invalid decimal value. Try again.
Enter any number between -100 and 100: -101
Error! Number must be greater than -101.0
Enter any number between -100 and 100: 101
Error! Number must be less than 101.0
Enter any number between -100 and 100: 100

Required String Test
Enter your email address:
Error! This entry is required. Try again.
Enter your email address: joelmurach@yahoo.com

String Choice Test
Select one (x/y):
Error! This entry is required. Try again.
Select one (x/y): q
Error! Entry must be 'x' or 'y'. Try again.
Select one (x/y): x
```

Operation

- This application prompts the user to enter a valid integer within a specified range, a valid double within a specified range, a required string, and one of two strings. If an entry isn't valid, the application displays an appropriate error message.

Specifications

- Create a class named OOValidator that contains these constructors and methods:

```
public OOValidator(Scanner sc)
public int getInt(String prompt)
public int getIntWithinRange(String prompt, int min, int max)
public double getDouble(String prompt)
public double getDoubleWithinRange(String prompt,
    double min, double max)
```

- Create a class named MyValidator that extends the OOValidator class. This class should add two new methods:

```
public String getRequiredString(String prompt)
public String getChoiceString(String prompt,
   String s1, String s2)
```

- Create a class named ValidatorTestApp and use it to test the methods in the Validator, OOValidator, and MyValidator classes.

Homework 4-2: Work with customer and employee data

Console

```
Welcome to the Person Tester application

Create customer or employee? (c/e): c

Enter first name: Frank
Enter last name: Jones
Enter email address: frank44@hotmail.com
Customer number: M10293

You entered:
Name: Frank Jones
Email: frank44@hotmail.com
Customer number: M10293

Continue? (y/n): y

Create customer or employee? (c/e): e

Enter first name: Anne
Enter last name: Prince
Enter email address: anne@murach.com
Social security number: 111-11-1111

You entered:
Name: Anne Prince
Email: anne@murach.com
Social security number: 111-11-1111

Continue? (y/n): n
```

Operation

- The application prompts the user to enter a customer or an employee.
- If the user selects customer, the application asks for name, email, and customer number.
- If the user selects employee, the application asks for name, email, and social security number.
- When the user finishes entering data for a customer or employee, the application displays the data that the user entered.

Specifications

- Create an abstract Person class that stores first name, last name, and email address. This class should provide a no-argument constructor, get and set methods for each piece of data, and it should override the toString method so it returns the first name, last name, and email fields in this format:

```
Name: Frank Jones
Email: frank44@hotmail.com
```

  In addition, it should contain an abstract method named getDisplayText that returns a string.

- Create a class named Customer that inherits the Person class. This class should store a customer number, it should provide get and set methods for the customer number, it should provide a no-argument constructor, and it should provide an implementation of the getDisplayText method. The getDisplayText method should return a string that consists of the string returned by the toString method of the Person class appended with the Customer number like this:

```
Name: Frank Jones
Email: frank44@hotmail.com
Customer number: M10293
```

- Create a class named Employee that inherits the Person class. This class should store a social security number, it should provide get and set methods for the social security number, it should provide a no-argument constructor, and it should provide an implementation of the getDisplayText method. The getDisplayText method should return a string that consists of the string returned by the toString method of the Person class appended with the Employees social security number like this:

```
Name: Anne Prince
Email: anne@murach.com
Social security number: 111-11-1111
```

- Create a class named PersonApp that prompts the user as shown in the console output. This class should create the necessary Customer and Employee objects from the data entered by the user, and it should use these objects to display the data to the user. To print the data for an object to the console, this application should use a static method named print that accepts a Person object.

- Use the Validator class or a variation of it to validate the user's entries.