CIT 149: Java I
Chapter 6 Lab 2

As the program become larger I will not include flowcharts. By this time you should be familiar with the processes within programs. This lab will complete programming project # 9 on page 468. This program will create a Person class for usability and a separate program to test the class.

PersonCh6

1. Open a new document and save the file as PersonCh6.java. First we will create block comments explaining the purpose. Type:

```
/*

File name: PersonCh6.java

Class for a person: an enhanced version of PersonImproved class
from Chapter 5.

Enhancements:
   Four constructors:
      1. with parameters for both attributes, name and age.
      2. with just a name parameter (sets age = 0).
      3. with just an age parameter (sets name = "No name").
      4. default constructor with no parameters (sets age = 0
         and name = "No name".

An object of this class has two attributes, a name and an age.
```

Public methods:

Set just the name, just the age, or both.

Test if two persons are equal (same name and age).

Test if two persons have the same name.

Test if two persons have the same age.

Test if one person is older than another.

Test if one person is younger than another.

Preconditions:

For set methods: the person has been created.

For test methods: the persons have been created and have values for name and age.

Postconditions:

Set methods: one or both attributes is assigned a value.

Test methods: true or false is returned.

2. Type the code that will import all classes within the java.util package.
3. Type the class header and opening brace.
4. Declare a String variable by the name of *name* and an integer named *age*. Make both variables private.
5. We will have 4 different constructor methods. This means that we will overload each, with each having different parameters. Type:

```
//The four constructors


 //#1 - with both name and age parameters.

 //     - sets age to 0 if a negative number is passed.
```

```
public PersonCh6(String newName, int newAge)
{
    name = newName;
    if(newAge >= 0)
        age = newAge;
    else
    {
        System.out.println("Age must be a non-negative number.");
        System.out.println("The age has been set to 0.");
        age = 0;
    }
}

//#2 - with just a name parameter.
//     - sets age to 0.
public PersonCh6(String newName)
{
    name = newName;
    age = 0;
}

//#3 - with just an age parameter.
//     - sets name to "No name".
//   - sets age = 0 if a negative number is passed.
public PersonCh6(int newAge)
```

```
{
    name = "No name";
    if(newAge >= 0)
        age = newAge;
    else
    {
        System.out.println("Age must be a non-negative number.");
        System.out.println("The age has been set to 0.");
        age = 0;
    }
}

//#4 - default constructor (no parameters).
//   - sets name to "No name".
//     - sets age to 0.
public PersonCh6()
{
    name = "No name";
    age = 0;
}
```

- You should be able to understand which each of these constructor methods do.

6. Our next method is the readInput() method. Type:

```
public void readInput()
 {
    Scanner keyboard = new Scanner(System.in);


    System.out.println("What is the person's name?");
    name = keyboard.nextLine();


    System.out.println("What is the person's age?");
    age = keyboard.nextInt();
    while (age < 0)
    {
       System.out.println("Age cannot be negative.");
       System.out.println("Reenter age:");
       age = keyboard.nextInt();
    }
 }
```

- This method constructs a Scanner object
- Asks the user to enter the name and age
- If the age is less than 0 a message will display and the user will be asked to reenter the age.

7. The writeOutput() method will display the name and age. Type:

```
public void writeOutput()
 {
    System.out.println("Name = " + name);
    System.out.println("Age = " + age);
 }
```

8. The set() method is a mutator method that will change the current value of the two variables. Type:

```
public void set(String newName, int newAge)
 {
    name = newName;
    if (newAge >= 0)
       age = newAge;
    else
    {
       System.out.println("ERROR: Used a negative age.");
       System.exit(0);
    }
 }
```

9. The setName() and setAge() methods are also mutator methods but they will handle the changing of only one of the variables. Type:

```
public void setName(String newName)
 {
    name = newName;
 }


 public void setAge(int newAge)
 {
    if (newAge >= 0)
       age = newAge;
    else
    {
       System.out.println("ERROR: Used a negative age.");
       System.exit(0);
    }
 }
```

10. The getName() and getAge() methods are accessor methods. They access the current values of the variables and return the value to where the method is invoked. Type:

```
public String getName()

 {

    return name;

 }


 public int getAge()

 {

    return age;

 }
```

11. The equals() method checks to see if the variables are equal to the values for the object passed to the method. The method will return true if they are equal, otherwise it will return false. Type:

```
public boolean equals(PersonCh6 another)

 {

    return(this.name.equalsIgnoreCase(another.name)

            && this.getAge() == another.getAge());

 }
```

12. The isSameName() and isSameAge() methods are similar to the equals() method except that they check only one of the variables. Type:

```
public boolean isSameName(PersonCh6 another)
 {
    return(this.name.equalsIgnoreCase(another.name));
 }


 public boolean isSameAge(PersonCh6 another)
 {
    return(this.getAge() == another.getAge());
 }
```

13. The next two methods check to see if the ages are older or younger. Type:

```
public boolean isOlderThan(PersonCh6 another)
 {
    return(this.getAge() > another.getAge());
 }


 public boolean isYoungerThan(PersonCh6 another)
 {
    return(this.getAge() < another.getAge());
 }
```

14. Close the class and compile the program. Fix any errors as necessary.

PersonCh6Test class

This class will test the PersonCh6 class using different scenarios. You should be able to understand what is occurring, but I will add explanations after different sections. I am only testing the PersonCh6 class 10 times, unlike in the instructions from the book.

1. Open a new document window and save the program as PersonCh6Test.java. First for our block comments that describe the purpose of the program. Type:

```
/**
*
* File name: PersonCh6Test.java
*
* A program to test the class PersonCh6.
*
* This program creates objects of class PersonCh6 and
* exercises its methods, including the four constructors.
* PersonCh6 objects have two data attributes, name and age.
*
* PersonCh6 public methods:
*         constructor with name and age parameters.
*         constructor with just name parameter (sets age = 0).
*         constructor with just age parameter (sets name = "No name").
*         default constructor (no parameters) - sets name = "No name"
*                 and age = 0.
*   readInput interactively assigns name and age values.
*   writeOutput displays name and age.
```

```
 *      set assigns name and age.

 *   getName returns name.

 *   setName sets name.

 *   setAge sets age.

 *      test if two persons are equal (same name and age)
*/
```

2. Type the import statement that will import the Scanner class within the java.util package.
3. Type the class header, opening brace and main method header and opening brace.
4. Construct a new Scanner object named keyboard.
5. Our first test will test the default values of *name* and *age* and display them. Type:

```
System.out.println();

System.out.println("Test case 1: default constructor and");

System.out.println("writeOutput() method.");

System.out.println();

PersonCh6 secretAgent1 = new PersonCh6();

System.out.println("Results of default constructor:");

System.out.println("Should be name = \"No name\" and");

System.out.println("age = 0.");

System.out.println();

secretAgent1.writeOutput();

System.out.println();

System.out.println("====================================");
```

- We construct a new PersonCh6 object using the default constructor method.
- and use the writeOutput() method to display the default values.

6. Our second test will test the readInput() method. Type:

```
  System.out.println("Test case 2: readInput() method.");
secretAgent1.readInput();
System.out.println();
System.out.println("Verify name and age.");
System.out.println("Should be whatever you just entered.");
System.out.println();
secretAgent1.writeOutput();
System.out.println();
System.out.println("======================================");
```

- Here we invoke the readInput() method and use the writeOutput() method to verify that the *name* and *age* variables changed values.

7. Next we construct a new PersonCh6 object using the constructor method that has parameters of only a String. Type:

```
System.out.println("Test case 3: new person that tests constructor with just name.");

   String case3Name;

   System.out.println("Enter a new name");

   case3Name = keyboard.next();

   PersonCh6 secretAgent2 = new PersonCh6(case3Name);

   System.out.println();

   System.out.println("Verify name = a new name and age = 0.");

   System.out.println();

   secretAgent2.writeOutput();

   System.out.println();

   System.out.println("====================================");
```

- I created a new String variable and requested that the user enter a name.
- I constructed a new Personch6 object passing the variable value to the constructor method that had parameters of a single String.

8. Our next test will change the age for the secretAgent2 object. Type:

```
System.out.println("Test case 4: constructor with just age.");
   System.out.println();
   System.out.println("Set the age = 40.");
   System.out.println();
   secretAgent2.setAge(40);
   secretAgent2.writeOutput();
   System.out.println();
   System.out.println("====================================");
```

- Using the setAge() method the value of age is changed. The writeOutput() method will display the new values.

9. The fifth test will request the user enter a new name and age and set the values for a new object. Type:

```
System.out.println("Test case 5: constructor with both name & age.");

   String case5Name;

   int case5Age;

   System.out.println("Enter a new name");

   case5Name = keyboard.next();

   System.out.println("What is their age?");

   case5Age = keyboard.nextInt();

   System.out.println();

   System.out.println("Verify new name and age");

   System.out.println();

   PersonCh6 secretAgent3 = new PersonCh6(case5Name, case5Age);

   secretAgent3.writeOutput();

   System.out.println();

   System.out.println("=====================================");
```

10. The sixth test verifies the name by invoking the getName() method. Type:

```
  System.out.println();

System.out.println("Test case 6:");

System.out.println("Test getName() method");

System.out.println();

System.out.println("Verify results: "+ secretAgent3.getName());

System.out.println("=====================================");
```

11. We test the getAge() method in the same way. Type:

```
System.out.println();
   System.out.println("Test case 7:");
   System.out.println("Test getAge() method");
   System.out.println();
   System.out.println("Verify results: "
                            + secretAgent3.getAge());
   System.out.println("====================================");
```

12. We change the name for secretAgent3. Type:

```
System.out.println();
   System.out.println("Test case 8:");
   System.out.println("Change the last name to Rocky");
   secretAgent3.setName("Rocky");
   System.out.println();
   System.out.println("Verify results with getName(): "
                            + secretAgent3.getName());
   System.out.println("====================================");
```

- The setName() method mutates the value of name.

13. We do the same for age. Type:

```
System.out.println();
   System.out.println("Test case 9:");
   System.out.println("setAge to 99");
   secretAgent3.setAge(99);
   System.out.println();
   System.out.println("Verify results: "
                              + secretAgent3.getAge());
   System.out.println("===================================");
```

14. Our final code changes the values for the first two objects and check to see if they're equal. Type:

```
System.out.println();

    System.out.println("Test case 10: set method (both name & age) on the first, two
  persons");

    System.out.println("and equals (both name and age)");

    System.out.println("Making two persons"

                                  + " with same name and age:");

    secretAgent1.set("Bullwinkle", 10);

    secretAgent2.set("Bullwinkle", 10);

    System.out.println("First person: ");

    secretAgent1.writeOutput();

    System.out.println("Second person: ");

    secretAgent2.writeOutput();

    System.out.println();

    System.out.println("Verify results: should be true.");

    System.out.println(secretAgent1.equals(secretAgent2));

    System.out.println("====================================");
```

- If secretAgent1 equals secretAgent2 it will return true.

15. Close the main method and class.
16. Compile the program, fix any errors if necessary and run it.
17. Compress ALL files into a single zip or rar file and submit.