Preguntas de revisión

- Preguntas en azul.
- Respuestas en verde.

1. The logical structure in which one instruction occurs after another with no branching is a:
   a. sequence
   b. selection
   c. loop
   d. case
2. Which of the following is typically used in a flowchart to indicate a decision?
   a. square
   b. rectangle
   c. diamond
   d. oval
3. Which of the following is not a type of if statement?
   a. single-alternative if
   b. dual-alternative if
   c. reverse if
   d. nested if
4. A decision is based on a(n) ____ value.
   a. Boolean
   b. absolute
   c. definitive
   d. convoluted
5. In Java, the value of (4 > 7) is:
   a. 4
   b. 7
   c. true
   d. false

6. Assuming the variable q has been assigned the value 3, which of the following statements displays XXX?

   a. if(q > 0) System.out.println("XXX");

   b. if(q > 7); System.out.println("XXX");

   c. Both of the above statements display XXX.

   d. Neither of the above statements displays XXX.

7. What is the output of the following code segment?

```
t = 10;
if(t > 7)
{
        System.out.print("AAA");
        System.out.print("BBB");
}
```

   a. AAA

   b. BBB

   c. AAABBB

   d. nothing

8. What is the output of the following code segment?

```
t = 10;
if(t > 7)
        System.out.print("AAA");
        System.out.print("BBB");
```

   a. AAA

   b. BBB

   c. AAABBB

   d. nothing

9. What is the output of the following code segment?

```
t = 7;
if(t > 7)
        System.out.print("AAA");
```

```
System.out.print("BBB");
```

a. AAA

b. BBB

c. AAABBB

d. nothing

10. When you code an if statement within another if statement, as in the following, then the if statements are:

```
if(a > b)
    if(c > d)    x = 0;
```

a. notched

b. nestled

c. nested

d. sheltered

11. The operator that combines two conditions into a single Boolean value that is true only when both of the conditions are true, but is false otherwise:

a. $$

b. !!

c. ||

d. &&

12. The operator that combines two conditions into a single Boolean value that is true when at least one of the conditions is true is:

a. $$

b. !!

c. ||

d. &&

13. Assuming a variable f has been initialized to 5, which of the following statements sets g to 0?

a. if(f > 6 || f == 5) g = 0;

b. if(f < 3 || f > 4) g = 0;

c. if(f >= 0 || f < 2) g = 0;

d. All of the above statements set g to 0.

14. Which of the following groups has the lowest operator precedence?

a. relational

b. equality

c. addition

d. logical OR

15. Which of the following statements correctly outputs the names of voters who live in district 6 and all voters who live in district 7?

a. if(district == 6 || 7)

```
System.out.println("Name is " + name);
```

b. if(district == 6 || district == 7)

```
System.out.println("Name is " + name);
```

c. if(district = 6 && district == 7)

```
System.out.println("Name is " + name);
```

d. two of these

16. Which of the following displays “Error” when a student ID is less than 1000 or more than 9999?

a. if(stuId < 1000) if(stuId > 9999)

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

b. if(stuId < 1000 && stuId > 9999)

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

c. if(stuId < 1000)

```
System.out.println("Error");
else
    if(stuId > 9999)
        System.out.println("Error");
```

d. Two of these are correct.

17. You can use _____ the statement to terminate a switch structure.

a. switch

b. end

c. case

d. break

18. The argument tested in a switch structure can be any of the following except a(n):

a. int

b. char

c. double

d. String

19. Assuming a variable w has been assigned the value 15, what does the following statement do?

w == 15 ? x = 2 : x = 0;

a. assigns 15 to w

b. assigns 2 to x

c. assigns 0 to x

d. nothing

20. Assuming a variable y has been assigned the value 6, the value of !(y < 7) is:

a. 6

b. 7

c. true

d. false

Rafael Antonio López García
Programación Orientada a Objetos

Ejercicios de programación

- Enlaces a los archivos en azul.

1. Write an application that asks a user to enter an integer. Display a statement that indicates whether the integer is even or odd. Save the file as **EvenOdd.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio1

2. Write an application that prompts the user for the day's high and low temperatures. If the high is greater than or equal to 90 degrees, display the message, "Heat warning." If the low is less than 32 degrees, display the message "Freeze warning." If the difference between the high and low temperatures is more than 40 degrees, display the message, "Large temperature swing." Save the file as **Temperatures.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio2

3. a. Write an application for the Summerdale Condo Sales office; the program determines the price of a condominium. Ask the user to choose *1* for park view, *2* for golf course view, or *3* for lake view. The output is the name of the chosen view as well as the price of the condo. Park view condos are $150,000, condos with golf course views are $170,000, and condos with lake views are $210,000. If the user enters an invalid code, set the price to 0. Save the file as **CondoSales.java**.

   b. Add a prompt to the `CondoSales` application to ask the user to specify a (1) garage or a (2) parking space, but only if the condo view selection is valid. Add $5,000 to the price of any condo with a garage. If the parking value is invalid, display an appropriate message and assume that the price is for a condo with no garage. Save the file as **CondoSales2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio3

4. a. The Williamsburg Women's Club offers scholarships to local high school students who meet any of several criteria. Write an application that prompts the user for a student's numeric high school grade point average (for example, 3.2), the student's number of extracurricular activities, and the student's number of service activities. Display the message "Scholarship candidate" if the student has any of the following:

   - A grade point average of 3.8 or above and at least one extracurricular activity and one service activity
   - A grade point average below 3.8 but at least 3.4 and a total of at least three extracurricular and service activities
   - A grade point average below 3.4 but at least 3.0 and at least two extracurricular activities and three service activities

   If the student does not meet any of the qualification criteria, display "Not a candidate." Save the file as **Scholarship.java**.

   b. Modify the `Scholarship` application so that if a user enters a grade point average under 0 or over 4.0, or a negative value for either of the activities, an error message appears. Save the file as **Scholarship2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio4

5. Write an application that displays a menu of three items for the Jivin' Java Coffee Shop as follows:

(1) American	1.99
(2) Espresso	2.50
(3) Latte	2.15

Prompt the user to choose an item using the number (1, 2, or 3) that corresponds to the item, or to enter *0* to quit the application. After the user makes the first selection, if the choice is *0*, display a total bill of $0. Otherwise, display the menu again. The user should respond to this prompt with another item number to order or *0* to quit. If the user types *0*, display the cost of the single requested item. If the user types *1*, *2*, or *3*, add the cost of the second item to the first, and then display the menu a third time. If the user types *0* to quit, display the total cost of the two items; otherwise, display the total for all three selections. Save the file as **Coffee.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio5

6. Barnhill Fastener Company runs a small factory. The company employs workers who are paid one of three hourly rates depending on skill level:

Skill Level	Hourly Pay Rate ($)
1	17.00
2	20.00
3	22.00

Each factory worker might work any number of hours per week; any hours over 40 are paid at one and one-half times the usual rate.

In addition, workers in skill levels 2 and 3 can elect the following insurance options:

Option	Explanation	Weekly Cost to Employee ($)
1	Medical insurance	32.50
2	Dental insurance	20.00
3	Long-term disability insurance	10.00

Also, workers in skill level 3 can elect to participate in the retirement plan at 3% of their gross pay.

Write an interactive Java payroll application that calculates the net pay for a factory worker. The program prompts the user for skill level and hours worked, as well as appropriate insurance and retirement options for the employee's skill level category. The application displays: (1) the hours worked, (2) the hourly pay rate, (3) the regular pay for 40 hours, (4) the overtime pay, (5) the total of regular and overtime pay, and (6) the total itemized deductions. If the deductions exceed the gross pay, display an error message; otherwise, calculate and display (7) the net pay after all the deductions have been subtracted from the gross. Save the file as **Pay.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio6

7. Create a class for Shutterbug's Camera Store, which is having a digital camera sale. The class is named `DigitalCamera`, and it contains data fields for a brand, the number of megapixels in the resolution, and price. Include a constructor that takes arguments for the brand and megapixels. If the megapixel parameter is greater than 10, the constructor sets it to 10. The sale price is set based on the resolution; any camera with 6 megapixels or fewer is $99, and all other cameras are $129. Also include a method that displays `DigitalCamera` details. Write an application named `TestDigitalCamera` in which you instantiate at least four objects, prompt the user for values for the camera brand and number of megapixels, and display all the values. Save the files as **DigitalCamera.java** and **TestDigitalCamera.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio7

8. Create a class for the Parks Department in Cloverdale. The class is named `Park`, and it contains a `String` field for the name of the park, an integer field for the size in acres, and four Boolean fields that hold whether the park has each of these features: picnic facilities, a tennis court, a playground, and a swimming pool. Include get and set methods for each field. Include code in the method that sets the number of acres and does not allow a negative number or a number over 400. Save the file as **Park.java**.

   Then, create a program with methods that do the following:

   - Accepts a `Park` parameter and returns a Boolean value that indicates whether the `Park` has both picnic facilities and a playground.
   - Accepts a `Park` parameter and four Boolean parameters that represent requests for the previously mentioned `Park` features. The method returns `true` if the `Park` has all the requested features.
   - Accepts a `Park` parameter and four Boolean parameters that represent requests for the previously mentioned `Park` features. The method returns `true` if the `Park` has exactly all the requested features and no others.
   - Accepts a `Park` parameter and returns the number of facilities that the `Park` features.
   - Accepts two `Park` parameters and returns the larger `Park`.

   Declare at least three `Park` objects, and demonstrate that all the methods work correctly. Save the program as **TestPark.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio8

9. a. Create a class named `Invoice` that holds an invoice number, balance due, and three fields representing the month, day, and year when the balance is due. Create a constructor that accepts values for all five data fields. Within the constructor, assign each argument to the appropriate field with the following exceptions:

      - If an invoice number is less than 1000, force the invoice number to 0.
      - If the month field is less than 1 or greater than 12, force the month field to 0.
      - If the day field is less than 1 or greater than 31, force the day field to 0.
      - If the year field is less than 2011 or greater than 2017, force the year field to 0.

   In the `Invoice` class, include a display method that displays all the fields on an `Invoice` object. Save the file as **Invoice.java**.

   b. Write an application containing a `main()` method that declares several `Invoice` objects, proving that all the statements in the constructor operate as specified. Save the file as **TestInvoice.java**.

c. Modify the constructor in the `Invoice` class so that the day is not greater than 31, 30, or 28, depending on the month. For example, if a user tries to create an invoice for April 31, force it to April 30. Also, if the month is invalid, and thus forced to 0, also force the day to 0. Save the modified `Invoice` class as **Invoice2.java**. Then modify the `TestInvoice` class to create `Invoice2` objects. Create enough objects to test every decision in the constructor. Save this file as **TestInvoice2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio9

10. Use the Web to locate the lyrics to the traditional song "The Twelve Days of Christmas." The song contains a list of gifts received for the holiday. The list is cumulative so that as each "day" passes, a new verse contains all the words of the previous verse, plus a new item. Write an application that displays the words to the song starting with any day the user enters. (*Hint*: Use a `switch` statement with `cases` in descending day order and without any `break` statements so that the lyrics for any day repeat all the lyrics for previous days.) Save the file as **TwelveDays.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/ejercicio10

Game Zone

- Enlaces a los archivos en azul.

1. In Chapter 1, you created a class called `RandomGuess`. In this game, players guess a number, the application generates a random number, and players determine whether they were correct. Now that you can make decisions, modify the application so it allows a player to enter a guess before the random number is displayed and then displays a message indicating whether the player's guess was correct, too high, or too low. Save the file as **RandomGuess2.java**. (After you finish the next chapter, you will be able to modify the application so that the user can continue to guess until the correct answer is entered.)

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/gameZone1

2. Create a lottery game application. Generate three random numbers (see Appendix D for help in doing so), each between 0 and 9. Allow the user to guess three numbers. Compare each of the user's guesses to the three random numbers and display a message that includes the user's guess, the randomly determined three-digit number, and the amount of money the user has won as follows:

Matching Numbers	Award ($)
Any one matching	10
Two matching	100
Three matching, not in order	1,000
Three matching in exact order	1,000,000
No matches	0

Make certain that your application accommodates repeating digits. For example, if a user guesses 1, 2, and 3, and the randomly generated digits are 1, 1, and 1, do not give the user credit for three correct guesses—just one. Save the file as **Lottery.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/gameZone2

3. In Chapter 3, you created a `Card` class. Modify the `Card` class so the `setValue()` method does not allow a `Card`'s value to be less than 1 or higher than 13. If the argument to `setValue()` is out of range, assign 1 to the `Card`'s value.

   In Chapter 3, you also created a `PickTwoCards` application that randomly selects two playing cards and displays their values. In that application, all `Card` objects arbitrarily were assigned a suit represented by a single character, but they could have different values, and the player observed which of two `Card` objects had the higher value. Now, modify the application so the suit and the value both are chosen randomly. Using two `Card` objects, play a very simple version of the card game War. Deal two `Cards`—one for the computer and one for the player—and determine the higher card, then display a message indicating whether the cards are equal, the computer won, or the player won. (Playing cards are considered equal when they have the same value, no matter what their suit is.) For this game, assume the Ace (value 1) is low. Make sure that the two `Cards` dealt are not the same `Card`. For example, a deck cannot contain more than one `Card` representing the 2 of spades.

If two cards are chosen to have the same value, change the suit for one of them. Save the application as **War.java**. (After you study the chapter *Arrays*, you will be able to create a more sophisticated War game in which you use an entire deck without repeating cards.)

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/gameZone3

4. In Chapter 4, you created a `Die` class from which you could instantiate an object containing a random value from 1 through 6. You also wrote an application that randomly "throws" two dice and displays their values. Modify the application so it determines whether the two dice are the same, the first has a higher value, or the second has a higher value. Save the application as **TwoDice2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/gameZone4

5. In the game Rock Paper Scissors, two players simultaneously choose one of three options: rock, paper, or scissors. If both players choose the same option, then the result is a tie. However, if they choose differently, the winner is determined as follows:

   - Rock beats scissors, because a rock can break a pair of scissors.
   - Scissors beats paper, because scissors can cut paper.
   - Paper beats rock, because a piece of paper can cover a rock.

   Create a game in which the computer randomly chooses rock, paper, or scissors. Let the user enter a number 1, 2, or 3, each representing one of the three choices. Then, determine the winner. Save the application as **RockPaperScissors.java**. (In the chapter *Characters, Strings, and the* `StringBuilder`, you will modify the game so that the user enters a string for *rock*, *paper*, and *scissors*, rather than just entering a number.)

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/gameZone5

Case Problems

- Enlaces a los archivos en azul.

1. a. Carly's Catering provides meals for parties and special events. In Chapters 3 and 4, you created an `Event` class for the company. Now, make the following changes to the class:
   - Currently, the class contains a field that holds the price for an `Event`. Now add another field that holds the price per guest, and add a `public` method to return its value.
   - Currently, the class contains a constant for the price per guest. Replace that field with two fields—a lower price per guest that is $32, and a higher price per guest that is $35.
   - Add a new method named `isLargeEvent()` that returns `true` if the number of guests is 50 or greater and otherwise returns `false`.
   - Modify the method that sets the number of guests so that a large `Event` (over 50 guests) uses the lower price per guest to set the new `pricePerGuest` field and calculate the total `Event` price. A small `Event` uses the higher price.

   Save the file as **Event.java**.

   b. In Chapter 4, you modified the `EventDemo` class to demonstrate two `Event` objects. Now, modify that class again as follows:
   - Instantiate three `Event` objects, and prompt the user for values for each object.
   - Change the method that displays `Event` details to use the new `isLargeEvent()` method and the new price per guest value. Use the display method with all three objects.
   - Create a method that accepts two `Event` objects and returns the larger one based on number of guests. (If the `Events` have the same number of guests, you can return either object.) Call this method three times—once with each pair of instantiated `Events`—and display the event number and number of guests for each argument as well as the event number and number of guests for the larger `Event`.

   Save the file as **EventDemo.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/caseProblem1

2. a. Sammy's Seashore Supplies rents beach equipment such as kayaks, canoes, beach chairs, and umbrellas to tourists. In Chapters 3 and 4, you created a `Rental` class for the company.

   Now, make the following change to the class:

   - Currently, a rental price is calculated as $40 per hour plus $1 for each minute over a full hour. This means that a customer who rents equipment for 41 or more minutes past an hour pays more than a customer who waits until the next hour to return the equipment. Change the price calculation so that a customer pays $40 for each full hour and $1 for each extra minute up to and including 40 minutes.

   Save the file as **Rental.java**.

   b. In Chapter 4, you modified the `RentalDemo` class to demonstrate a `Rental` object. Now, modify that class again as follows:

   - Instantiate three `Rental` objects, and prompt the user for values for each object. Display the details for each object to verify that the new price calculation works correctly.
   - Create a method that accepts two `Rental` objects and returns the one with the longer rental time. (If the `Rentals` have the same time, you can return either object.) Call this method three times—once with each pair of instantiated `Rentals`—and display the contract number and time in hours and minutes for each argument as well as the contract number of the longer `Rental`.

   Save the file as **RentalDemo.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo5/tree/main/chapter5_Joyce/src/caseProblem2