# Preguntas de revisión

- Preguntas en azul.
- Respuestas en verde.

1. In Java, methods must include all the following except:
    a. a declaration
    b. a call to another method
    c. curly braces
    d. a body
2. All method declarations contain:
    a. the keyword static
    b. one or more explicitly named access specifiers.
    c. arguments
    d. parentheses
3. A public static method named computeSum() is located in classA. To call the method from within classB, use the statement:
    a. computeSum(classB);
    b. classB(computeSum());
    c. classA.computeSum();
    d. You cannot call computeSum() from within classB.
4. Which of the following method declarations is correct for a static method named displayFacts() if the method receives an int argument?
    a. public static int displayFacts()
    b. public void displayFacts(int data)
    c. public static void displayFacts(int data)
    d. Two of these are correct.
5. The method with the declaration public static int aMethod(double d) has a method type of:
    a. static
    b. int
    c. double

d. You cannot determine the method type.

6. Which of the following is a correct call to a method declared as public static void aMethod(char code)?

   a. void aMethod();

   b. void aMethod('V');

   c. aMethod(char 'M');

   d. aMethod('Q');

7. A method is declared as public static void showResults(double d, int i).

   Which of the following is a correct method call?

   a. showResults(double d, int i);

   b. showResults(12.2, 67);

   c. showResults(4, 99.7);

   d. Two of these are correct.

8. The method with the declaration public static char procedure(double d) has a method type of:

   a. public

   b. static

   c. char

   d. double

9. The method public static boolean testValue(int response) returns:

   a. a boolean value

   b. an int value

   c. no value

   d. You cannot determine what is returned.

10. Which of the following could be the last legally coded line of a method declared as public static int getVal(double sum)?

    a. return;

    b. return 77;

    c. return 2.3;

    d. Any of these could be the last coded line of the method.

11. The nonstatic data components of a class often are referred to as the of that class:
    a. access types
    b. instance variables
    c. methods
    d. objects
12. Objects contain methods and data items, which are also known as:
    a. fields
    b. functions
    c. themes
    d. instances
13. You send messages or information to an object through its:
    a. fields
    b. methods
    c. classes
    d. type
14. A program or class that instantiates objects of another prewritten class is a(n):
    a. class client
    b. superclass
    c. object
    d. patron
15. The body of a class is always written:
    a. in a single line, as the first statement in a class
    b. within parentheses
    c. between curly braces
    d. as a method call

16. Most class data fields are:
    a. private
    b. public
    c. static
    d. final
17. The concept of allowing a class's private data to be changed only by a class's own methods is known as:
    a. structured logic
    b. object orientation
    c. information hiding
    d. data masking
18. Suppose you declare an object as Book thisBook; Before you store data in thisBook, you:
    a. also must explicitly allocate memory for it
    b. need not explicitly allocate memory for it
    c. must explicitly allocate memory for it only if it has a constructor
    d. can declare it to use no memory
19. If a class is named Student, the class constructor name is:
    a. any legal Java identifier
    b. any legal Java identifier that begins with S
    c. StudentConstructor
    d. Student
20. If you use the automatically supplied default constructor when you create an object:
    a. numeric fields are set to 0 (zero)
    b. character fields are set to blank
    c. Boolean fields are set to true
    d. All of these are true

Rafael Antonio López García
Programación Orientada a Objetos

# Ejercicios de programación

- Respuestas en verde.
- Enlaces a los archivos en azul.

1. Suppose that you have created a program with only the following variables:

```
int v = 4;
int w = 6;
double x = 2.2;
```

Suppose that you also have a method with the following header:

```
public static void calculate(int x, double y)
```

Which of the following method calls are legal?

a. calculate(v, w);

b. calculate(v, x);

c. calculate(x, y);

d. calculate(18, x);

e. calculate(1.1, 2.2);

f. calculate(5, 7

2. Suppose that a class named `ClassA` contains a private nonstatic integer named `b`, a public nonstatic integer named `c`, and a public static integer named `d`. Which of the following are legal statements in a class named `ClassB` that has instantiated an object as `ClassA obA = new ClassA();`?

a. obA.b = 12;

b. obA.c = 5;

c. obA.d = 23;

d. ClassA.b = 4;

e. ClassA.c = 33;

f. ClassA.d = 99;

3. a. Create an application named `ArithmeticMethods` whose `main()` method holds two integer variables. Assign values to the variables. In turn, pass each value to methods named `displayNumberPlus10()`, `displayNumberPlus100()`, and `displayNumberPlus1000()`. Create each method to perform the task its name implies. Save the application as **ArithmeticMethods.java**.

   b. Modify the `ArithmeticMethods` class to accept the values of the two integers from a user at the keyboard. Save the file as **ArithmeticMethods2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio3

4. a. Create an application named `Percentages` whose `main()` method holds two `double` variables. Assign values to the variables. Pass both variables to a method named `computePercent()` that displays the two values and the value of the first number as a percentage of the second one. For example, if the numbers are 2.0 and 5.0, the method should display a statement similar to "2.0 is 40% of 5.0." Then call the method a second time, passing the values in reverse order. Save the application as **Percentages.java**.

   b. Modify the `Percentages` class to accept the values of the two `doubles` from a user at the keyboard. Save the file as **Percentages2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio4

5. When gasoline is $100 per barrel, then the consumer's price at the pump is between $3.50 and $4.00 per gallon. Create a class named `GasPrices`. Its `main()` method holds an integer variable named `pricePerBarrel` to which you will assign a value entered by a user at the keyboard. Create a method to which you pass `pricePerBarrel`. The method displays the range of possible prices per gallon. For example, if gas is $120 per barrel, then the price at the pump should be between $4.20 and $4.80. Save the application as **GasPrices.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio5

6. There are 2.54 centimeters in an inch, and there are 3.7854 liters in a U.S. gallon. Create a class named `MetricConversion`. Its `main()` method accepts an integer value from a user at the keyboard, and in turn passes the entered value to two methods. One converts the value from inches to centimeters and the other converts the same value from gallons to liters. Each method displays the results with appropriate explanation. Save the application as **MetricConversion.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio6

7. Assume that a gallon of paint covers about 350 square feet of wall space. Create an application with a `main()` method that prompts the user for the length, width, and height of a rectangular room. Pass these three values to a method that does the following:
   - Calculates the wall area for a room
   - Passes the calculated wall area to another method that calculates and returns the number of gallons of paint needed
   - Displays the number of gallons needed
   - Computes the price based on a paint price of $32 per gallon, assuming that the painter can buy any fraction of a gallon of paint at the same price as a whole gallon
   - Returns the price to the `main()` method

   The `main()` method displays the final price. For example, the cost to paint a 15- by-20-foot room with 10-foot ceilings is $64. Save the application as **PaintCalculator.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio7

8. The Harrison Group Life Insurance company computes annual policy premiums based on the age the customer turns in the current calendar year. The premium is computed by taking the decade of the customer's age, adding 15 to it, and multiplying by 20. For example, a 34-year-old would pay $360, which is calculated by adding the decades (3) to 15, and then multiplying by 20. Write an application that prompts a user for the current year and a birth year. Pass both to a method that calculates and returns the premium amount, and then display the returned amount. Save the application as **Insurance.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio8

9. Write an application that calculates and displays the weekly salary for an employee. The `main()` method prompts the user for an hourly pay rate, regular hours, and overtime hours. Create a separate method to calculate overtime pay, which is regular hours times the pay rate plus overtime hours times 1.5 times the pay rate; return the result to the `main()` method to be displayed. Save the program as **Salary.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio9

10. Write an application that calculates and displays the amount of money a user would have if his or her money could be invested at 5 percent interest for one year. Create a method that prompts the user for the starting value of the investment and returns it to the calling program. Call a separate method to do the calculation, and return the result to be displayed. Save the program as **Interest.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio10

11. a. Create a class named `Sandwich`. Data fields include a `String` for the main ingredient (such as "tuna"), a `String` for bread type (such as "wheat"), and a `double` for price (such as 4.99). Include methods to get and set values for each of these fields. Save the class as **Sandwich.java**.

    b. Create an application named `TestSandwich` that instantiates one `Sandwich` object and demonstrates the use of the set and get methods. Save this application as **TestSandwich.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio11

12. a. Create a class named `Student`. A `Student` has fields for an ID number, number of credit hours earned, and number of points earned. (For example, many schools compute grade point averages based on a scale of 4, so a three-credit-hour class in which a student earns an A is worth 12 points.) Include methods to assign values to all fields. A `Student` also has a field for grade point average. Include a method to compute the grade point average field by dividing points by credit hours earned. Write methods to display the values in each `Student` field. Save this class as **Student.java**.

    b. Write a class named `ShowStudent` that instantiates a `Student` object from the class you created and assign values to its fields. Compute the `Student` grade point average, and then display all the values associated with the `Student`. Save the application as **ShowStudent.java**.

    c. Create a constructor for the `Student` class you created. The constructor should initialize each `Student`'s ID number to 9999, his or her points earned to 12, and credit hours to 3 (resulting in a grade point average of 4.0). Write a program that demonstrates that the constructor works by instantiating an object and displaying the initial values. Save the application as **ShowStudent2.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio12

13. a. Create a class named `BankAccount` with fields that hold an account number, the owner's name, and the account balance. Include a constructor that initializes each field to appropriate default values. Also include methods to get and set each of the fields. Include a method named `deductMonthlyFee()` that reduces the balance by $4.00. Include a static method named `explainAccountPolicy()` that explains that the $4 service fee will be deducted each month. Save the class as **BankAccount.java**.

    b. Create a class named `TestBankAccount` whose `main()` method declares four `BankAccount` objects. Call a `getData()` method three times. Within the method, prompt a user for values for each field for a `BankAccount`, and return a `BankAccount` object to the `main()` method where it is assigned to one of `main()`'s `BankAccount` objects. Do not prompt the user for values for the fourth `BankAccount` object, but let it continue to hold the default values. Then, in `main()`, pass each `BankAccount` object in turn to a `showValues()` method that displays the data, calls the method that deducts the monthly fee, and displays the balance again. The `showValues()` method also calls the method that explains the deduction policy. Save the application as **TestBankAccount.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio13

14. a. Create a class named `Painting` that contains fields for a painting's title, artist, medium (such as water color), price, and gallery commission. Create a constructor that initializes each field to an appropriate default value, and create instance methods that get and set the fields for title, artist, medium, and price. The gallery commission field cannot be set from outside the class; it is computed as 20 percent of the price each time the price is set. Save the class as **Painting.java**.

    b. Create a class named `TestPainting` whose `main()` method declares three `Painting` items. Create a method that prompts the user for and accepts values for two of the `Painting` objects, and leave the third with the default values supplied by the constructor. Then display each completed object. Finally, display a message that explains the gallery commission rate. Save the application as **TestPainting.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/ejercicio14

# Game Zone

- Enlaces a los archivos en azul.

1. Playing cards are used in many computer games, including versions of such classics as solitaire, hearts, and poker. Design a `Card` class that contains a character data field to hold a suit (*s* for spades, *h* for hearts, *d* for diamonds, or *c* for clubs) and an integer data field for a value from 1 to 13. (When you learn more about string handling in the chapter *Characters, Strings, and the* `StringBuilder`, you can modify the class to hold words for the suits, such as *spades* or *hearts*, as well as words for some of the values—for example, *ace* or *king*.) Include get and set methods for each field. Save the class as **Card.java**.

   Write an application that randomly selects two playing cards and displays their values. Simply assign a suit to each of the cards, but generate a random number for each card's value. Appendix D contains information on generating random numbers. To fully understand the process, you must learn more about Java classes and methods. However, for now, you can copy the following statements to generate a random number between 1 and 13 and assign it to a variable:

   ```
   final int CARDS_IN_SUIT = 13;
   myValue = ((int)(Math.random() * 100) % CARDS_IN_SUIT + 1);
   ```

   After reading the chapter *Making Decisions*, you will be able to have the game determine the higher card. For now, just observe how the card values change as you execute the program multiple times. Save the application as **PickTwoCards.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/gameZone1

2. Computer games often contain different characters or creatures. For example, you might design a game in which alien beings possess specific characteristics such as color, number of eyes, or number of lives. Design a character for a game, creating a class to hold at least three attributes for the character. Include methods to get and set each of the character's attributes. Save the file as **MyCharacter.java**. Then write an application in which you create at least two characters. In turn, pass each character to a display method that displays the character's attributes. Save the application as **TwoCharacters.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/gameZone2

# Case Problems

- Enlaces a los archivos en azul.

1. a. Carly's Catering provides meals for parties and special events. In Chapter 2, you wrote an application that prompts the user for the number of guests attending an event, displays the company motto with a border, and then displays the price of the event and whether the event is a large one. Now modify the program so that the `main()` method contains only three executable statements that each call a method as follows:

   - The first executable statement calls a `public static int` method that prompts the user for the number of guests and returns the value to the `main()` method.
   - The second executable statement calls a `public static void` method that displays the company motto with the border.
   - The last executable statement passes the number of guests to a `public static void` method that computes the price of the event, displays the price, and displays whether the event is a large event.

   Save the file as **CarlysEventPriceWithMethods.java**.

   b. Create a class to hold `Event` data for Carly's Catering. The class contains:

   - Two `public final static` fields that hold the price per guest ($35) and the cutoff value for a large event (50 guests)
   - Three `private` fields that hold an event number, number of guests for the event, and the price. The event number is stored as a `String` because Carly plans to assign event numbers such as *M312*.
   - Two `public` set methods that set the event number (`setEventNumber()`) and the number of guests (`setGuests()`). The price does not have a set method because the `setGuests()` method will calculate the price as the number of guests multiplied by the price per guest every time the number of guests is set.
   - Three `public` get methods that return the values in the three nonstatic fields

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

   c. Use the `CarlysEventPriceWithMethods` class you created in Step 1a as a starting point for a program that demonstrates the `Event` class you created in Step 1b, but make the following changes:

   - You already have a method that gets a number of guests from a user; now add a method that gets an event number. The `main()` method should declare an `Event` object, call the two data entry methods, and use their returned values to set the fields in the `Event` object.
   - Call the method from the `CarlysEventPriceWithMethods` class that displays the company motto with the border. The method is accessible because it is public, but you must fully qualify the name because it is in another class.
   - Revise the method that displays the event details so that it accepts the newly created `Event` object. The method should display the event number, and it should still display the number of guests, the price per guest, the total price, and whether the event is a large event.

   Save the program as **EventDemo.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/caseProblem1

2. a. Sammy's Seashore Supplies rents beach equipment such as kayaks, canoes, beach chairs, and umbrellas to tourists. In Chapter 2, you wrote an application that prompts the user for the number of minutes a piece of sports equipment was rented, displays the company motto with a border, and displays the price for the rental. Now modify the program so that the `main()` method contains only three executable statements that each call a method as follows:

   - The first executable statement calls a method that prompts the user for the rental time in minutes and returns the value to the `main()` method.
   - The second executable statement calls a method that displays the company motto with the border.
   - The last executable statement passes the number of minutes to a method that computes the hours, extra minutes, and price for the rental, and then displays all the details.

   Save the file as **SammysRentalPriceWithMethods.java**.

   b. Create a class to hold `Rental` data for Sammy's Seashore Supplies. The class contains:

   - Two `public final static` fields that hold the number of minutes in an hour and the hourly rental rate ($40)
   - Four `private` fields that hold a contract number, number of hours for the rental, number of minutes over an hour, and the price. The contract number is stored as a `String` because Sammy plans to assign contract numbers such as *K681*.
   - Two `public` set methods. One sets the contract number (`setContractNumber()`). The other is named `setHoursAndMinutes()`, and it accepts the number of minutes for the rental and then sets the hours, extra minutes over an hour, and the total price. Recall from Chapter 2 that the price is $40 per hour plus $1 for every extra minute.
   - Four `public` get methods that return the values in the four nonstatic fields

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

   c. Use the `SammysRentalPriceWithMethods` class you created in Step 2a as a starting point for a program that demonstrates the `Rental` class you created in Step 2b, but make the following changes:

   - You already have a method that gets a number of minutes from a user; now add a method that gets a contract number. The `main()` method should declare a `Rental` object, call the two data entry methods, and use their returned values to set the fields in the `Rental` object.
   - From the `SammysRentalPriceWithMethods` class, call the `RentalDemo` method that displays the company motto with the border. The method is accessible because it is public, but you must fully qualify the name because it is in another class.
   - Revise the method that displays the rental details so that it accepts the newly created `Rental` object. The method should display the contract number, and it should still display the hours and minutes, the hourly rate, and the total price.

   Save the program as **RentalDemo.java**.

Enlace al archivo:
https://github.com/logralahad/POO1_Capitulo3/tree/main/chapter3_Joyce/src/caseProblem2