class Person

  # Step 4: Define the initializer method

  def initialize(name, age, email)

    @name = name

    @age = age

    @email = email

  end

  # Step 5: Getter methods to retrieve attribute values

  def get\_name

    @name

  end

  def get\_age

    @age

  end

  def get\_email

    @email

  end

  # Step 6: Setter methods to modify attribute values

  def set\_name(name)

    @name = name

  end

  def set\_age(age)

    @age = age

  end

  def set\_email(email)

    @email = email

  end

end

# Create a new person object

person1 = Person.new("John Doe", 30, "john.doe@example.com")

# Retrieve and display the person's information

puts "Name: #{person1.get\_name}"

puts "Age: #{person1.get\_age}"

puts "Email: #{person1.get\_email}"

# Modify the person's information

person1.set\_name("Jane Doe")

person1.set\_age(28)

person1.set\_email("jane.doe@example.com")

# Display the updated information

puts "Updated Name: #{person1.get\_name}"

puts "Updated Age: #{person1.get\_age}"

puts "Updated Email: #{person1.get\_email}"

ruby person.rb

Ruby is a versatile and user-friendly programming language that excels in web development, particularly with the Ruby on Rails framework. Its emphasis on readability and simplicity makes it a popular choice for beginners and experienced developers alike.

This Ruby code defines a class `Person` and demonstrates how to create and manipulate an instance of that class. Let’s break it down step by step.

### 1. Class Definition

```ruby

class Person

```

- This line defines a new class called `Person`. In Ruby, classes are used to create objects that can hold data and define behaviors.

### 2. Initializer Method

```ruby

def initialize(name, age, email)

@name = name

@age = age

@email = email

end

```

- \*\*`initialize` Method\*\*: This is a special method in Ruby that is called when you create a new instance of the class (an object). It sets up the initial state of the object.

- \*\*Instance Variables\*\*: The `@name`, `@age`, and `@email` variables are instance variables that belong to the object. They store the values passed when creating a `Person` object.

### 3. Getter Methods

```ruby

def get\_name

@name

end

def get\_age

@age

end

def get\_email

@email

end

```

- These methods are called \*\*getter methods\*\*. They allow access to the private instance variables from outside the class.

- When you call `person1.get\_name`, it returns the value of the `@name` variable for that instance.

### 4. Setter Methods

```ruby

def set\_name(name)

@name = name

end

def set\_age(age)

@age = age

end

def set\_email(email)

@email = email

end

```

- These methods are called \*\*setter methods\*\*. They allow you to modify the values of the private instance variables.

- For example, calling `person1.set\_name("Jane Doe")` changes the `@name` variable for that instance to "Jane Doe".

### 5. Creating an Instance of the Class

```ruby

person1 = Person.new("John Doe", 30, "john.doe@example.com")

```

- This line creates a new `Person` object named `person1` using the `new` method, which calls the `initialize` method with the provided arguments.

- The `@name`, `@age`, and `@email` instance variables are set to "John Doe", 30, and "john.doe@example.com", respectively.

### 6. Retrieving Information

```ruby

puts "Name: #{person1.get\_name}"

puts "Age: #{person1.get\_age}"

puts "Email: #{person1.get\_email}"

```

- These lines use the getter methods to retrieve the values of the instance variables for `person1` and print them to the console.

- The output will be:

```

Name: John Doe

Age: 30

Email: john.doe@example.com

```

### 7. Modifying Information

```ruby

person1.set\_name("Jane Doe")

person1.set\_age(28)

person1.set\_email("jane.doe@example.com")

```

- Here, the setter methods are used to update the instance variables of `person1` to new values.

### 8. Displaying Updated Information

```ruby

puts "Updated Name: #{person1.get\_name}"

puts "Updated Age: #{person1.get\_age}"

puts "Updated Email: #{person1.get\_email}"

```

- These lines again use the getter methods to retrieve and print the updated values of the instance variables.

- The output will be:

```

Updated Name: Jane Doe

Updated Age: 28

Updated Email: jane.doe@example.com

```

### Summary

- The code defines a `Person` class with an initializer, getter methods, and setter methods.

- An instance of `Person` is created with initial values.

- The code demonstrates how to retrieve and modify the object's attributes using getter and setter methods.

- This encapsulation of data (using instance variables) and controlled access/modification (using methods) is a key concept in object-oriented programming, promoting clean and maintainable code.