Макарова Анастасія КС31

Контрольна робота. Теоретична частина

Варіант 3

1. Що таке об'єктно-орієнтоване програмування, і як Ruby підтримує його?
2. Які основні принципи наслідування та поліморфізму в об'єктно-орієнтованому програмуванні, і як Ruby їх реалізує?

Питання 1:

Об’єктно-орієнтоване програмування (ООП) - це підхід до програмування, де основними елементами є об’єкти. Об’єкти містять дані та методи для роботи з цими даними.

У ООП є такі ключові концепції як:

- **Класи та об’єкти**: Клас - це шаблон або креслення для створення об’єктів. Об’єкт - це екземпляр класу.
- **Інкапсуляція**: Дані об’єкта приховані від зовнішнього світу, і доступ до них здійснюється через методи об’єкта.
- **Наслідування**: Клас може успадковувати властивості та методи іншого класу.
- **Поліморфізм**: Об’єкти можуть приймати багато форм в залежності від контексту.

Ruby - це мова програмування, яка повністю підтримує ООП. У Ruby все є об’єктами, навіть примітивні типи даних. Ruby також підтримує динамічну типизацію, що дозволяє змінювати типи даних під час виконання програми.

Приклад коду на Ruby:

```
class Animal
  def initialize(name)
    @name = name
  end

  def speak
    "#{@name} *звук*."
  end
end

#Dog успадковує властивості класу Animal
class Dog < Animal
  def speak
    "#{@name} гавкає!"  #перевизначення метода speak для Dog
  end
end

#Cat успадковує властивості класу Animal
class Cat < Animal
  def speak
    "#{@name} мурчить!"
  end
```

```
end
dog = Dog.new("Mr. Pickles")
cat = Cat.new("Tom")
#кожний об'єкт виконує цей метод по-своєму.
puts dog.speak
puts cat.speak
```

Animal - це базовий клас, а Dog і Cat - це підкласи, які успадковують властивості класу Animal і перевизначають метод speak. Це демонструє принципи ООП на прикладі Ruby. На відміну від деяких інших мов, таких як C++ або Java, Ruby не має явних абстрактних класів або інтерфейсів. Замість цього він використовує модулі для надання повторно використовуваного коду. Це робить Ruby дуже гнучким для об'єктно-орієнтованого програмування.

Питання 2:

Об'єктно-орієнтоване програмування (ООП) включає в себе кілька ключових принципів, таких як наслідування та поліморфізм. В Ruby ці принципи реалізуються з допомогою специфічних механізмів:

Наслідування в ООП - це механізм, який дозволяє створювати нові класи на основі вже існуючих, успадковуючи їх властивості та поведінку. У Ruby наслідування реалізується як концепція ООП, що дозволяє новим класам використовувати властивості існуючого класу. Однак у Ruby підтримується лише одиночне наслідування, тобто кожен клас може мати лише один батьківський клас. Багатократне наслідування заборонено в Ruby. Замість цього використовуються модулі (MIXIN), які дозволяють імітувати багатократне наслідування. Модулі в Ruby - це збірки методів і констант, які можна включити в будь-який клас. Це забезпечує гнучкість та ефективність наслідування в Ruby, зберігаючи при цьому простоту і зрозумілість коду.

Поліморфізм в ООП дозволяє об'єктам обробляти специфічні для них запити та вести себе по-різному в залежності від типу об'єкта. У Ruby підтримується два основних види поліморфізму:

- Наслідуваний поліморфізм: Це найбільш поширена форма поліморфізму, коли підклас успадковує методи від свого суперкласу. Підклас може перевизначити ці методи для створення більш спеціалізованої поведінки. У мовах із статичною типізацією, таких як C++, наслідуваний поліморфізм також забезпечує сумісність типів.
- Інтерфейсний поліморфізм: Цей вид поліморфізму не вимагає наслідування між класами. Замість цього, різні класи можуть реалізовувати один і той же інтерфейс, який визначає набір методів.

У Ruby підтримується інтерфейсний поліморфізм за допомогою використання модулів. Модулі в Ruby - це колекції методів і констант, які можна "підмішати" до існуючих класів. Це дозволяє класам використовувати методи модуля так, немов би вони були частиною самого класу, забезпечуючи деяку форму множинного наслідування.

Важливо зауважити, що Ruby - мова з динамічною типізацією, що означає, що типи об'єктів визначаються під час виконання програми, а не під час компіляції. Це дозволяє Ruby

забезпечити поліморфізм без необхідності статичної типізації. Таким чином, Ruby підтримує обидва основні принципи наслідування та поліморфізму в об'єктно-орієнтованому програмуванні, з використанням одиночного наслідування та модулів для досягнення багатократного наслідування та інтерфейсного поліморфізму.