Контрольна робота
Студента групи КС32
**Коханчука Юрія**
З дисципліни МПП
**Варіант 3**

1.

**DSL, або "Domain-Specific Language"** (Мова Специфічної Області), представляє собою мову програмування чи конфігурації, яка була спроектована для конкретної області або завдання. Основна ідея DSL полягає в тому, щоб мати мову, яка була б природною та легко зрозумілою для фахівців у певній області та, можливо, спрощувала би задачі чи давала би можливість компілювати програму по-іншому для окремих задач.
Основні характеристики DSL:

- **Орієнтованість на конкретну область:**
  DSL спеціально розроблені для вирішення конкретних задач в певній області, наприклад, конфігурація системи, опис бізнес-правил, опис графічних інтерфейсів тощо.

**Високорівневі або низькорівневі:**
DSL може бути високорівневим (спрощений та орієнтованим на людей) або низькорівневим (близьким до мови програмування, оптимізованим для комп'ютерного розуміння).

- **Простота та зрозумілість:**
  DSL часто створюється так, щоб його легко було зрозуміти фахівцям у конкретній області, навіть якщо вони не є програмістами.

**Приклади різних DSL можуть включати:**

- Конфігураційні файли:

**Мови запитів:**

- SQL

**Мови графічного програмування**

- GLSL
- HLSL

2.
В Ruby, **include** та **extend** - це два ключові слова, які використовуються для роботи з модулями.

**include:**
Коли ви включаєте модуль в клас за допомогою **include**, методи модуля стають екземплярними методами класу. Це означає, що вони будуть доступні для екземплярів класу. При включенні модуля в клас його методи надаються екземплярам класу, а не самому класу. include зазвичай використовується для розширення функціональності екземплярів класу.

```
module MyModule
  def my_method
    puts "This is a method from MyModule"
  end
end

class MyClass
  include MyModule
end

obj = MyClass.new
obj.my_method
```

У цьому прикладі метод **my_method** з модуля **MyModule** тепер доступний для об'єкту **obj**.

**extend:**
Коли ви розширюєте клас або об'єкт за допомогою **extend**, методи модуля стають класовими методами (якщо ви розширюєте клас) або методами об'єкта (якщо ви розширюєте конкретний об'єкт). Методи модуля стають доступними без створення екземплярів класу чи об'єкта. **extend** зазвичай використовується для розширення функціональності класу чи конкретного об'єкта.

```
module MyModule
  def my_method
    puts "This is a method from MyModule"
  end
end

class MyClass
  extend MyModule
end

MyClass.my_method
```

3.

```
require 'thread'


$array = [0,1,2,3,4,5,6,7,8,9,10]
# Створення масиву
$tmp = 0
# лічільник
def fun
  mutex = Mutex.new
  while $tmp<10
# блокування ресурсу мьютеком, створення потоку та друк
    th1 = Thread.new{mutex.lock; print "TH1 "; puts $array[$tmp]; $tmp+=1;
mutex.unlock}
# блокування ресурсу мьютеком, створення потоку та друк
    th2 = Thread.new{mutex.lock; print "TH2 "; puts $array[$tmp]; $tmp+=1;
mutex.unlock}
    th1.join
    th2.join
# приєдання потоків
  end
end

fun
```

```
TH1 0
TH2 1
TH1 2
TH2 3
TH1 4
TH2 5
TH1 6
TH2 7
TH1 8
TH2 9
```

**Рис.1 – виконання програми.**