Теорія мов програмування

Лабораторна робота 4. Імплементація ООП

4.1 Деяка програма була написана та відкомпільована. Нижче приведене повідомлення про помилку від компілятора. Що ви можете помітити з повідомлення компіляції? Чому це важливо для Java?

```
test.java:1: error: class Test is public,
   should be declared in a file named Test.java
public class Test {
       ^
1 error
My Mac>
```

4.2 Використовуючи C ++, ми не маємо вимог до імен для модулів та класів, таких як в Java. Отже, коли клас **A** використовує клас **Test**, і клас **A**, і клас **Test** можуть бути поміщені у файл з будь-яким ім'ям. Чому це нормально для програм на C ++, але не для програм на Java?

4.3 Аргументи командного рядка вводяться після назви програми. Наприклад, якщо програма називається **grep**, ви можете надати їй такі аргументи командного рядка:

```
grep def *. py
```

Як програми C ++, так і Java можуть отримувати аргументи командного рядка через основну функцію. З C ++ кількість аргументів командного рядка передається як **argc**, а фактичні аргументи передаються в масив рядків (тобто масиви символів) у параметрі з іменем **argv**, як зазначено на рис. 7.9. Змінна **argc** завжди є принаймні однією для програм на C ++, але довжина аргументів командного рядка Масив рядків у Java може бути нульовим, якщо аргументи командного рядка не передаються. Ти знаєш чому?

4.4 Ми бачили, як поліморфізм забезпечується мовами програмування C ++ та Java. Поліморфізм також забезпечується Python. Проте з Python ми не оголошуємо методи віртуальними, як в C ++, і ми не маємо вбудованої ієрархії класів, як Java. Як відбувається поліморфізм у Python?

4.5 Java **ArrayList** містить два перевантажені методи, які називаються remove. Береться параметр int і видаляється об'єкт із зазначеним індексом у списку **ArrayList**. Інший приймає об'єкт як параметр і видаляє перший екземпляр об'єкта з **ArrayList**. Чому це може створити проблему?

4.6 Існування класу з іменем **PySuper** передбачає, що класи можна будувати динамічно (тобто під час виконання). Потреба в **PySuper** випливає з

необхідності динамічного пошуку суперкласу під час роботи програми, оскільки загалом вона не може бути відома до її використання. Яка інструкція в додатку А відповідає за підготовку JCoCo до динамічного створення класу?

4.7 Які вільні змінні та зв’язані змінні в цій функції Python?

```
def f(x):
    y = x
    return aVal + lstInts[0] + y
```

4.8 Розглянемо код у прикладі нижче.

```
public PyClass(String name, ArrayList<PyObject> nestedclassesandfuns,
            String baseClass, HashMap<String, PyObject> globals)  {
    super(name, PyTypeId.PyClassType);
    this.baseClass = baseClass;
    this.name = name;
    this.classesandfuns = nestedclassesandfuns;
    this.globals = (HashMap<String, PyObject>)globals;
    this.attrs.put("__name__", new PyStr(name));
    for (int i = 0; i < classesandfuns.size(); i++) {
        if (classesandfuns.get(i).getType().typeId() == PyTypeId.PyCodeType
            PyCode code = (PyCode) classesandfuns.get(i);
            ArrayList<PyObject> env = new ArrayList<PyObject>();
            for (int j = 0; j<code.getFreeVars().size(); j++) {
                String freeVar = code.getFreeVars().get(j);
                if (freeVar.equals("__class__")) {
                    env.add(new PyCell(this));
                } else {
                    throw new PyException(ExceptionType.PYMATCHEXCEPTION,
                     "Error: Found unexpected freevar in class declaration.
                }
            }
            PyFunction fun = new PyFunction((PyCode) classesandfuns.get(i),
                globals, new PyTuple(env));
            this.attrs.put(fun.callName(), fun);
        } else if (classesandfuns.get(i).getType().typeId() ==
                  PyTypeId.PyTypeType) {
            PyClass cls = (PyClass) classesandfuns.get(i);
            this.attrs.put(cls.getName(), cls);
        } else {
            throw new PyException(ExceptionType.PYMATCHEXCEPTION,
                       "TypeError: expected a Function or Class, got "+
                       classesandfuns.get(i).getType().str());
        }
    }
}
public void initInstance(PyObjectAdapter obj)  {
    if (!baseClass.equals("")) {
        ((PyClass)globals.get(baseClass)).initInstance(obj);
    }
    for (String name : this.attrs.keySet()) {
        if (this.attrs.get(name).getType().typeId() ==
              PyTypeId.PyFunctionType) {
            obj.dict.put(name, new PyMethod(name, obj,
                (PyCallable)this.attrs.get(name)));
        }
    }
}
@Override
public PyObject __call__(ArrayList<PyObject> args)  {
    PyObjectAdapter obj = new PyObjectInst(this);
    initInstance(obj);
    ((PyMethod) obj.dict.get("__init__")).__call__(args);
    return obj;
}
```

Коли створюється об'єкт **Dog**, його метод **_ _init _ _** неявно викликається. Ми ніколи не викликаємо конструктор на об'єкті Python. Де **_ _init _ _** викликається у віртуальній машині JCoCo?

```
mydog = Dog ("Mesa")
```

4.9 Як об’єкт або клас можуть замінити поведінку за замовчуванням магічного методу, такого як метод **_ _str _ _**, не змінюючи саму віртуальну машину JcoCo?

Виконані завдання надішліть на пошту викладачеві.

Контрольні запитання

1. Що означає перевірка статичного типу? Чи є в C ++? Чи є це у Python? Чи є у Java?
2. Які назви та цілі двох програм, що складають середовище Java для виконання програм?
3. Яка проблема номер один, з якою повинні мати справу програми C / C ++? Чому це не проблема для програм Java та Python?
4. Що робить інструмент **make** і як він працює для програм на C ++?
5. Чи існує еквівалент інструменту **make** для програм Java?
6. Як компілятор C ++ розрізняє директиви процесорів макросів та оператори C / C ++?
7. Що таке простір імен у C ++? Що можна порівняти з простором імен у Java? У Python?
8. Яке виконуване ім’я за замовчуванням для скомпільованої програми на C ++?
9. Що таке окрема компіляція і чому вона важлива?
10. Що таке динамічне зв’язування? Це відбувається в C ++ або в Java? Чому це важливо?
11. В якому середовищі вбудовано збір сміття, C ++ або Java?
12. Які переваги збору сміття?
13. Чи є недоліки у вивозі сміття?
14. Що таке деструктор і коли він потрібен?
15. Що потрібно написати, щоб отримати поліморфний метод у C ++?
16. Яка мета поліморфізму?
17. Яка мета спадкування?
18. Чим відрізняються інтерфейси та класи в Java? Чим вони схожі? Чим вони відрізняються?
19. Що таке клас адаптера? Чому вони корисні?
20. Що таке зворотний дзвінок і як вони зазвичай реалізуються в Java?
21. Що таке дженерики? Чому вони зручні?
22. Що таке шаблон? Як оголосити вектор у C ++?
23. Що таке автобокс та розпакування?

24. Як функція представлена як значення у Java?
25. Що таке анонімний клас?
26. Що таке тип (6) у JCoCo та Python? Як щодо типу (тип (6))? Як щодо типу (тип (тип (тип (6)))? Чому не цікаво йти далі?
27. Сканер JCoCo базується на кінцевому автоматі. Як реалізований кінцевий автомат? Які основні конструкції використовуються кінцевим станом машина?
28. Чи працює парсер JCoCo знизу вгору або зверху вниз?
29. Як у JCoCo пов’язані об’єкт **`PyCode`** та об’єкт **`PyFunction`**?
30. Що таке відстеження і чому це важливо?
31. Яка мета класу **`PyMethod`**?
32. Прибуття хеш-значень для хеш-об'єктів на Java є тривіальним. Опишіть, як JCoCo визначає хеш-значення для об’єктів у реалізації об’єктів **`PyDict`**.