МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет “Львівська політехніка”

Кафедра САПР

Розрахункова робота

На тему:

“Ознайомлення із фреймворком JCCLEC для еволюційних обчислень.”

з курсу “Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні”

Варіант 6

Виконала:

ст. гр. КНс-13

Супрун Н. П.

Прийняв:

Кривий Р.З

Львів – 2017

**Мета**

Ознайомитись із фреймворками для роботи з еволюційними обчислення та генетичним програмуванням.

**Індивідуальне завдання (Варіант 6)**

Ознайомитись із фреймворком **JCLEC** для роботи із еволюційними обчисленнями, навести короткі теоретичні відомості про його загальні характеристики, описати процес встановлення, навести кілька прикладів роботи та отримані у ході їх виконання результати.

**Теоретичні відомості**

***JCLEC*** – програмне забезпечення для еволюційних обчислень (*англ. Evolutionaty Computation (EC)*), розроблене на мові програмування Java. Надає фреймворк високого рівня для виконання будь якого еволюційного алгоритму, забезпечуючи при цьому пітримку генетичних алгоритмів (із бінарним, цілочисельним та дробовим кодуванням), генетичного програмування (стиль Кози, строга типізація, граматична орієнтовність) та еволюційного програмування.

Архітектура JCLEC дотримується принципів об’єктно-орієнтованого програмування, де абстракції представлені парними об’єктами і де повторне використання коду є типовим і простим. JCLEC забезпечує середовище, яке можна охарактеризувати такими особливостями:

* **Універсальність**. Будь-який еволюційний алгоритм може бути виконаний за допомогою JCLEC за умови, що дотримано певні мінімальні вимоги. Єдина необхідна умова – мати популяцію індивідів, до якої буде ітеративно застосована послідовність еволюційних операцій. На даний момент JCLEC підтримує найбільш популярні особливості еволюційного програмування, серед яких генетичне програмування, бітові стрічки, генетичні алгоритмі на базі векторів із цілочисельними та дробовими векторами а також стратегію еволюції. Також є підтримка таких технік еволюційного програмування як мультиоб’єктна оптимізація. Користувач може скористатися будь-яким із цих спеціалізованих фреймворків, чи навіть змінити їх та створити власний користувацький еволюційний алгоритм.
* **Інтуїтивність**. Приклавши суттєві зусилля, JCLEC вдалось побудувати у простій та приємній формі. JCLEC забезпечує різноманітні алгоритми, супроводжені інтуїтивно зрозумілим інтерфейсом. Крім того, стиль програмування є високорівневим, що дозволяє швидке прототипіювання застосунків.
* **Портативність**. JCLEC було розроблено на мові програмуванні Java, тож система працювати на усіх платформах, що підтримують Java Virtual Machine.
* **Ефективність**. Щоб переконатись у ефективності, неабияку увагу було приділено критичним фрагментам коду. Кожен фрагмент було ретельно проаналізовано на предмет швидкодії.
* **Надійність**. Твердження верифікації та валідації були добавлені у код щоб переконатись, що всі операції валідні, і що користувач отримує звіт про проблему у разі її виникнення.
* **Елегантність**. Інтерфейс JCLEC розроблювався дуже ретельно. Багато сил було докладено для того, щоб розробити когерентний програмний пакунок, за яким стоять найкращі принципи об’єктно-орієнтованого програмування та програмування в принципі. Більше того, були дотримані строгі правила програмування, які зобов’язували писати код так, щоб його було зручно читати, розуміти та, рано чи пізно, вдосконалювати. До центральних аспектів JCLEC належить також використання файлів формату XML, що забезпечує загальні підходи до розробки інструментів, аналізу та генерації файлів, інтеграції фреймворку в інші системи.
* **Відкритий сирцевий код**. Сирцевий код JCLEC знаходиться у відкритому доступі та доступний для використання за умови дотримання ліцензії GNU General Public License (GPL). Його можна розповсюджувати та вдосконалювати абсолютно безкоштовно.

Програму можна поділити на чотири великі модулі, кожен із яких також має свої особливості.

**Загальні особливості:**

* Здатність легко розширювати функціонал та повторно його використовувати (згідно із принципами ООП);
* Налаштування алгоритму задаються із зовнішнього файла;
* Підтримка багатопоточності;
* Наявність кількох генераторів випадкових чисел:
  + Ranecu;
  + Ranmar;
  + Ranlux;
  + Ranmt;
* Модульне виконання алгоритму;
* Графічний інтерфейс із можливістю виведення графіків.

**Особливості еволюційнийх обчислень:**

* **Підтримка кількох кодувань:**
  + Лінійний генотип: бінарне, цілочисельне та дробовочисельне кодування;
  + Девероподібний генотип: дерева виразів;
  + Нейронні мережі: багатошаровий перцептрон, радіальні базисні мережі.
* **Бібліотека еволюційних алгоритмів містить:**
  + Класичні еволюційні алгоритми;
  + Мультиобєктні алгоритми;
  + Меметичні алгоритми.

**Особливості генетичних алгоритмів:**

* **Лінійне бінарне кодування:**
  + Оператори переходу: одноточкові, двоточкові, рівномірний перехід, HUX-перехід;
  + Оператори мутації: один локус, кілька локусів та рівномірний мутатор.
* **Лінійне цілочисельне кодування:**
  + Оператори переходу: одноточкові, двоточкові, рівномірний перехід, HUX-перехід;
  + Оператори мутації: один локус, кілька локусів та рівномірний мутатор.
* **Лінійне цілочисельне кодування:**
  + Оператори переходу: BLX-альфа, арифметичний, рівномірний 2х1, рівномірний 2х2 і т. д.
  + Оператори мутації: випадкова, нерівномірна, модальна дискретна, модальна неперервна і т. д.

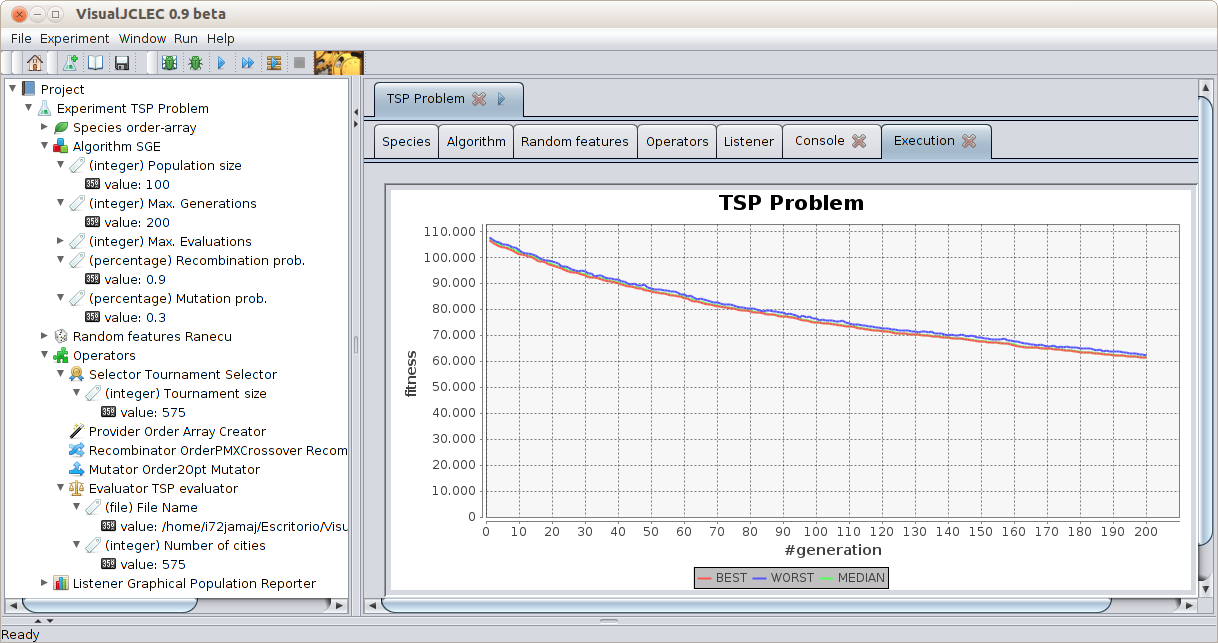
**Особливості генетичного програмування:**

* **Кодування дерева виразів (Koza та строго типізоване генетичне програмування):**
  + Оператори переходу: переходи дерева та підддерева;
  + Оператори мутації: піддерево, одна ланка, всі ланки.
* **Кодування синтаксичного дерева (граматично кероване генетичне програмування):**
  + Оператори переходу: переходи дерева та вибірки;
  + Оператори мутації: рекурсивна, нерекурсивна, вибіркова, звужуюча, розширююча.

**VisualJCLEC**

Не дивлячись на те, що еволюційні обчислення є однім із найбільш часто використовуваних методів вирішення оптимізаційних задач, їх використання потребує глибоких знань у програмуванні, що перешкоджає використанню програми не-експертами. Тому потрібні механізми, які дозволять людям, які не є спеціалістами у програмування, користуватись засобами еволюційного програмування, наданими програмою. У таких ситуаціях часто хорошим рішенням є створення графічного середовища, яке дозволило ба налаштовувати параметри, виконувати алгоритми та візуалізовувати результати. Для реалізації графічного середовища було використано інструмент GenLab. Однак, у нього є суттєвий недолік: можливість маніпулювати лише фіксованим набором параметрів.

***VisualJCLEC*** – це адаптивний фреймворк JCLEC для еволюційних обчислень, який було розроблено у першу чергу для того, щоб вирішувати вищеописані проблеми. Щоб досягнути бажаного результату, гнучкість та масштабованість було обрано основними не-функціональними вимогами щодо дизайну інтерфейсу для VisualJCLEC. З одного боку, перший аспект полягає у здатності інструмент підлаштовувати функціонал під конкретні вимоги користувача. Більше того, під гнучкістю часто також розуміють здатність приймати нові підходи (наприклад, види, генетичні оператори, алгоритми, тощо) у репозиторіях, якими керує система. З іншого боку, масштабністю називають здатність інструменту вдосконалюватись за рахунок впровадження нового функціоналу.



**Рис 1.** Інтерфейс VisualJCLEC

**Встановлення**

Модулі JCLEC та його сирцевий код можна завантажити за посиланнями:

* Завантажити сирцевий код з [SourceForge](https://sourceforge.net/projects/jclec/files/4.0.0/)
* Завантажити сирцевий код з [SVN](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=20&Itemid=9)
* Завантажити сирцевий код з [GIT](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=21&Itemid=9)
* Завантажити сирцевий код з [CVS](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=15&Itemid=9)
* Завантажити сирцевий код з та імпортувати як проект у [Eclipse](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=16&Itemid=9)
* Завантажити сирцевий код з та імпортувати як проект у [NetBeans](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=17&Itemid=9)

Джерелом знань у роботі з JCLEC послужать:

* [Документація](http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=14&Itemid=7)
* [JCLEC wiki](http://jclec.sourceforge.net/mediawiki/index.php/JCLEC_wiki)
* [Туторіал JCLEC](http://jclec.sourceforge.net/data/JCLEC-tutorial.pdf)
* [Класифікаціний туторіал JCLEC](http://jclec.sourceforge.net/data/JCLEC-classification.pdf)

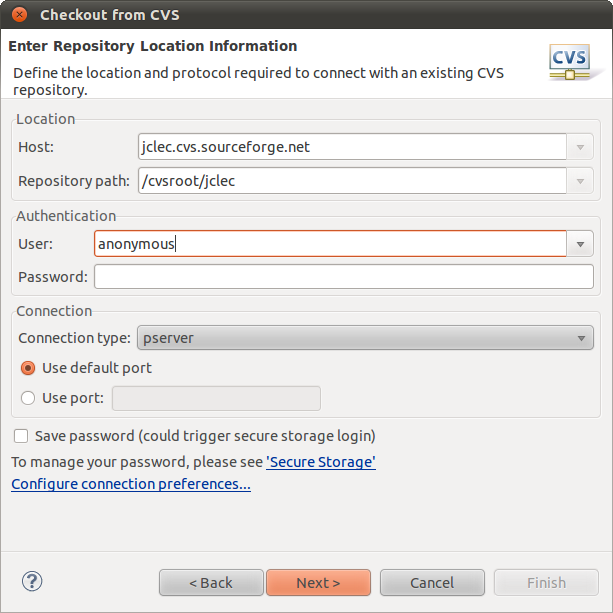
**Імпорт проекту в Eclipse**

Завантажити та імпортувати проект можна безпосередньо за допомогою середовища Eclipse.

1. File -> New -> Project
2. CVS -> Project from CVS

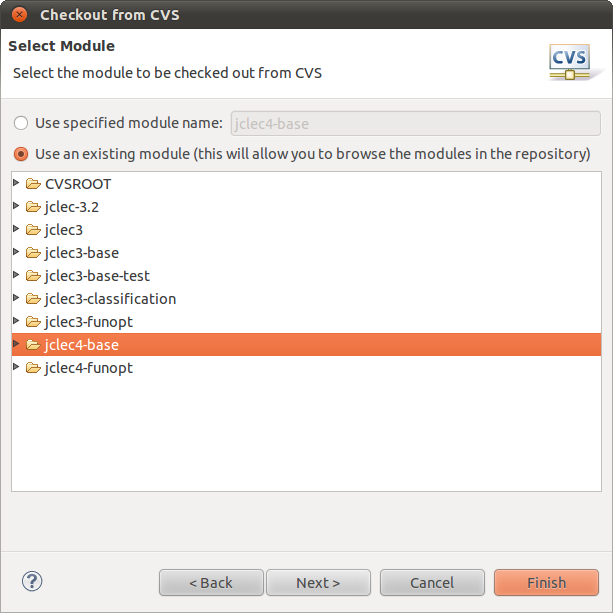
Заповніть форму такими даними:

|  |  |
| --- | --- |
| **Host:** | jclec.cvs.sourceforge.net |
| **Repository path:** | /cvsroot/jclec |
| **User:** | anonymous |



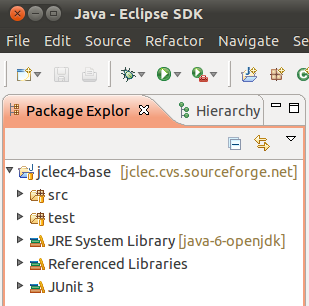
**Рис 2.** Встановлення проекту для Eclipse через CVS [1]

Оберіть пункт «**use an existing module**» та оберіть **jclec4-base**, **jclec4-classification** чи інші бажані модулі.



**Рис 3.** Встановлення проекту для Eclipse через CVS [2]

Тепер сирцевий код проектів jclec4-base та jclec4-classification, а також їхніх бібліотек імпортовано до вашого робочого простору.

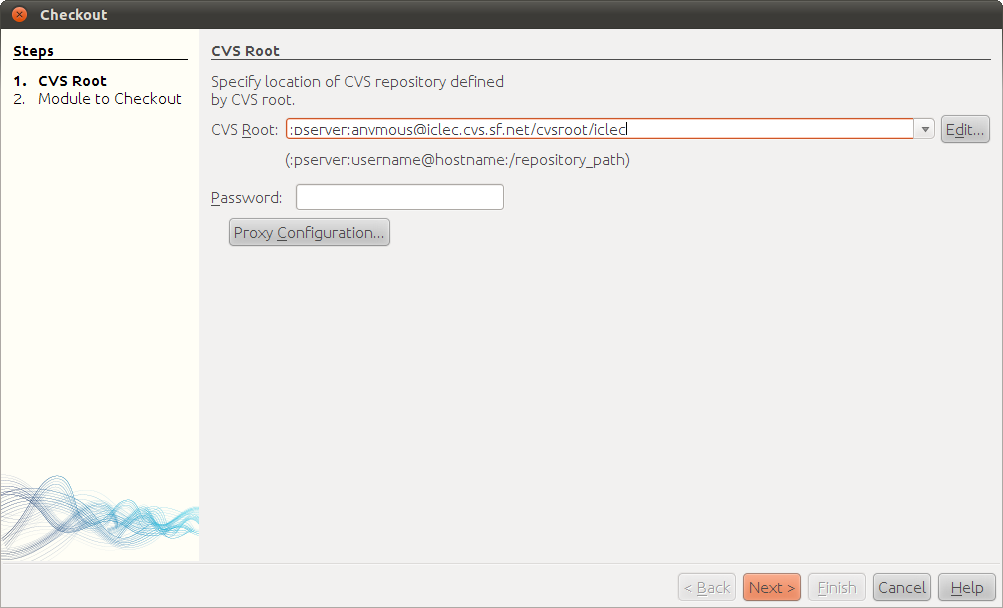


**Рис 4.** Встановлення проекту для Eclipse через CVS [3]

**Імпорт проекту в NetBeans**

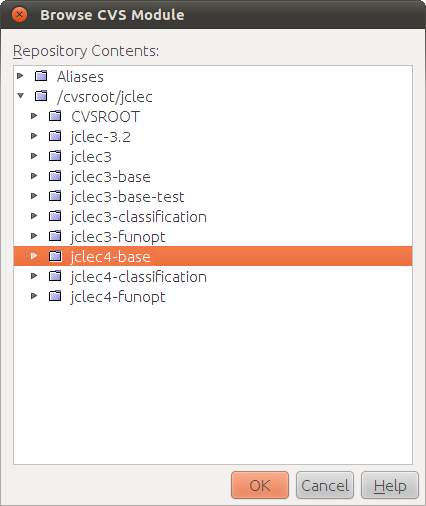
JCLEC та його модулі можна завантажити та імпортувати як NetBeans проекти. Для цього потрібно:

1. Натиснути Team -> CVS -> Checkout.
2. Заповнити форму таким чином:



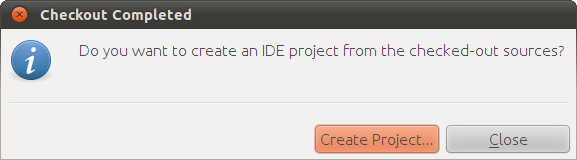
**Рис 5.** Встановлення проекту для NetBeans через CVS [1]

Переглянути наявні модулі CVS та обрати **jclec4-base**, **jclec4-classification** чи інші бажані модулі.

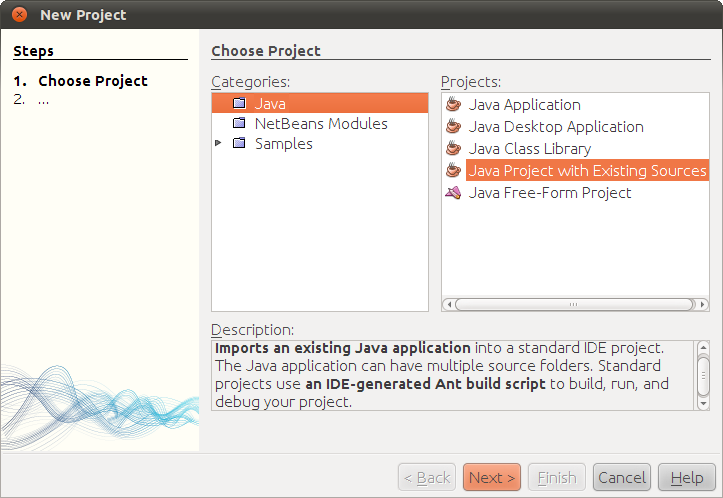


**Рис 6.** Встановлення проекту для NetBeans через CVS [2]

4. Обрати пункт Create Project.

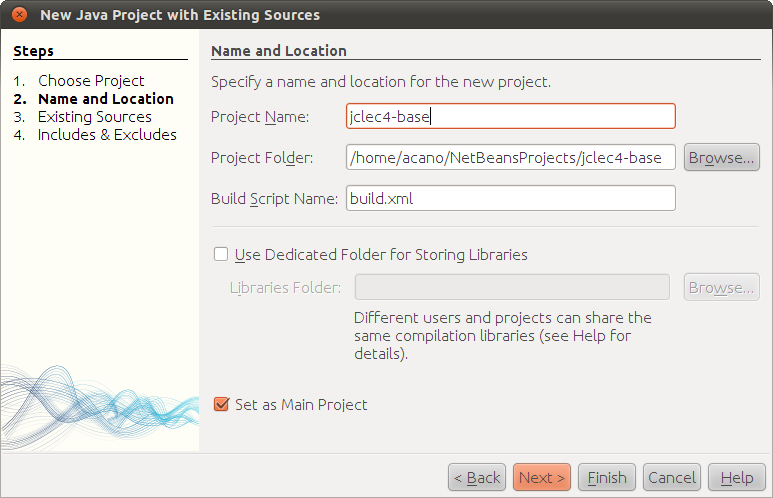


**Рис 7.** Встановлення проекту для NetBeans через CVS [3]



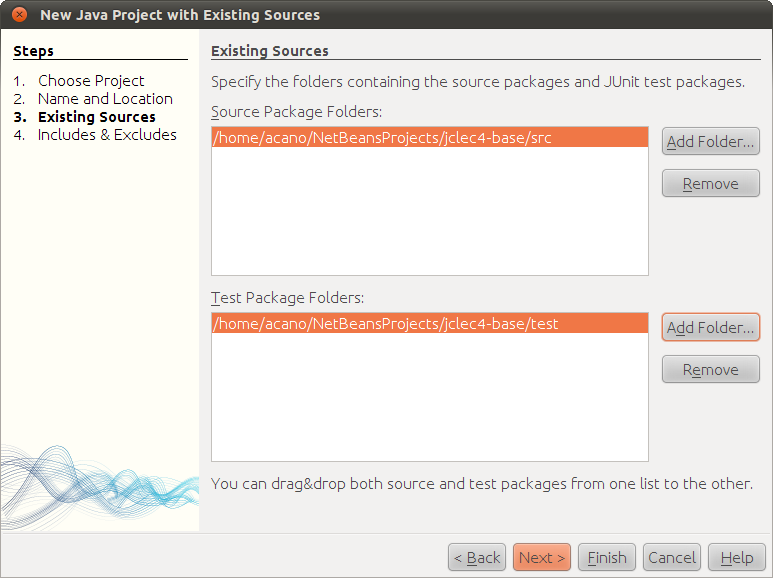
**Рис 8.** Встановлення проекту для NetBeans через CVS [4]

5. Назвати проекти **jclec4-base** та **jclec4-classification.**



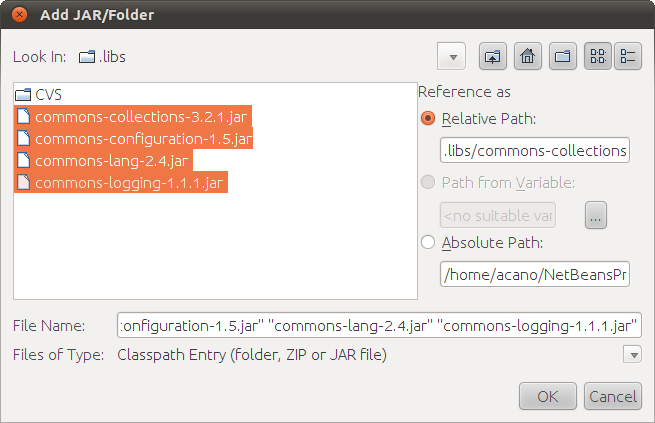
**Рис 9.** Встановлення проекту для NetBeans через CVS [5]

6. Визначити папки Source та Test.



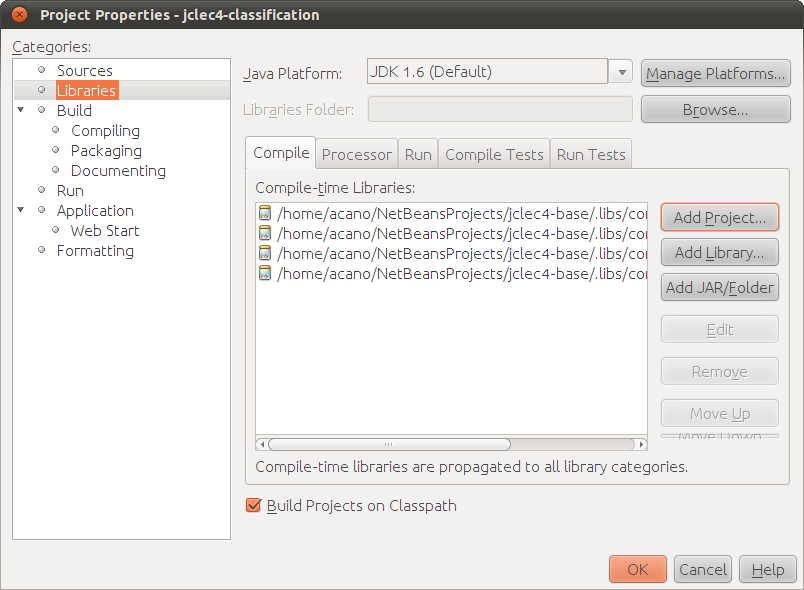
**Рис 10.** Встановлення проекту для NetBeans через CVS [6]

7. Додати необхідні бібліотеки, натиснувши **Add JAR\Folder** та обрати jar-файли зі папки libs.

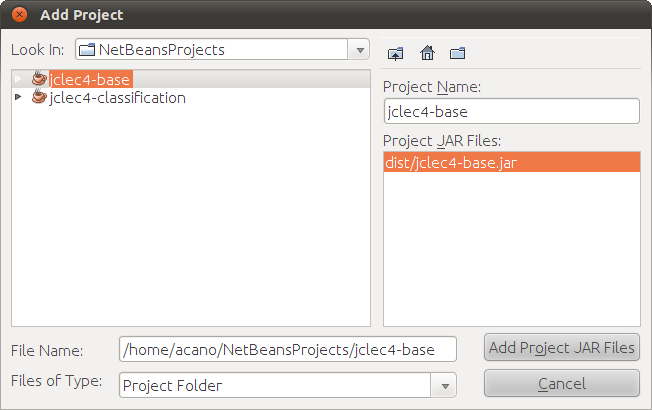


**Рис 11.** Встановлення проекту для NetBeans через CVS [7]

8. Для імпорту модулю **jclec4-classification** також додати модуль **jclec4-base**.



**Рис 12.** Встановлення проекту для NetBeans через CVS [8]



**Рис 13.** Встановлення проекту для NetBeans через CVS [9]

Готово! Тепер ви можете використовувати алгоритми із **jclec4-base** та **jclec4-classification**!

**Приклад використання фреймворку**

Наведений приклад вирішує задачу символьної регресії. Мета символічної регресії – отримати символьне представлення функцїі. Даний фрагмент програмного коду демонструє, як можна наблизити функції та

**X.java:**

import net.sf.jclec.exprtree.fun.Argument;

public class X extends Argument<Double>

{

public X()

{

super(Double.class, 0);

}

public boolean equals(Object other)

{

return other instanceof X;

}

public String toString()

{

return "X";

}

}

**Add.java:**

import net.sf.jclec.exprtree.fun.AbstractPrimitive;

import net.sf.jclec.exprtree.fun.ExprTreeFunction;

public class Add extends AbstractPrimitive

{

public Add()

{

super(new Class<?> [] {Double.class, Double.class}, Double.class);

}

@Override

protected void evaluate(ExprTreeFunction context)

{

// Get arguments (in context stack)

Double arg1 = pop(context);

Double arg2 = pop(context);

// Push result in context stack

push(context, arg1+arg2);

}

// java.lang.Object methods

public boolean equals(Object other)

{

return other instanceof Add;

}

public String toString()

{

return "+";

}

}

**Sub.java:**

import net.sf.jclec.exprtree.fun.AbstractPrimitive;

import net.sf.jclec.exprtree.fun.ExprTreeFunction;

public class Sub extends AbstractPrimitive

{

public Sub()

{

super(new Class<?> [] {Double.class, Double.class}, Double.class);

}

@Override

protected void evaluate(ExprTreeFunction context)

{

// Get arguments (in context stack)

Double arg1 = pop(context);

Double arg2 = pop(context);

// Push result in context stack

push(context, arg1-arg2);

}

// java.lang.Object methods

public boolean equals(Object other)

{

return other instanceof Sub;

}

public String toString()

{

return "−";

}

}

**Mul.java**

import net.sf.jclec.exprtree.fun.AbstractPrimitive;

import net.sf.jclec.exprtree.fun.ExprTreeFunction;

public class Mul extends AbstractPrimitive

{

public Mul()

{

super(new Class<?> [] {Double.class, Double.class}, Double.class);

}

@Override

protected void evaluate(ExprTreeFunction context)

{

// Get arguments (in context stack)

Double arg1 = pop(context);

Double arg2 = pop(context);

// Push result in context stack

push(context, arg1-arg2);

}

// java.lang.Object methods

public boolean equals(Object other)

{

return other instanceof Mul;

}

public String toString()

{

return "\*";

}

}

**SymregEvaluator.java:**

import java. util .Comparator;

import net.sf . jclec .IFitness;

import net.sf . jclec .IIndividual ;

import net.sf . jclec .base.AbstractEvaluator;

import net.sf . jclec .exprtree.ExprTree;

import net.sf . jclec .exprtree.ExprTreeIndividual;

import net.sf . jclec .exprtree.fun.ExprTreeFunction;

import net.sf . jclec . fitness .SimpleValueFitness;

import net.sf . jclec . fitness .ValueFitnessComparator;

public class SymregEvaluator extends AbstractEvaluator

{

private static final long *serialVersionUID* = 1L;

private static final Comparator<IFitness> *COMPARATOR* = new ValueFitnessComparator(true);

private static final double [] *xvalues* = {-2., -1., 0., 1., 2.};

private static final double [] *yvalues* = {10., 0., 0., 4., 30.};

@Override

protected void evaluate(IIndividual ind)

{

ExprTree genotype = ((ExprTreeIndividual) ind).getGenotype();

ExprTreeFunction function = new ExprTreeFunction(genotype);

double [] y = new double[*xvalues*.length];

for(int i = 0; i<*xvalues*.length; i++)

y[i ] = function.<Double>execute(*xvalues*[i]);

double rms = 0.0;

for (int i=0; i<*yvalues*.length; i++)

{

double diff = y[i] - *yvalues*[i ];

rms += diff\*diff ;

}

rms = Math.*sqrt*(rms);

ind.setFitness(new SimpleValueFitness(rms));

}

@Override

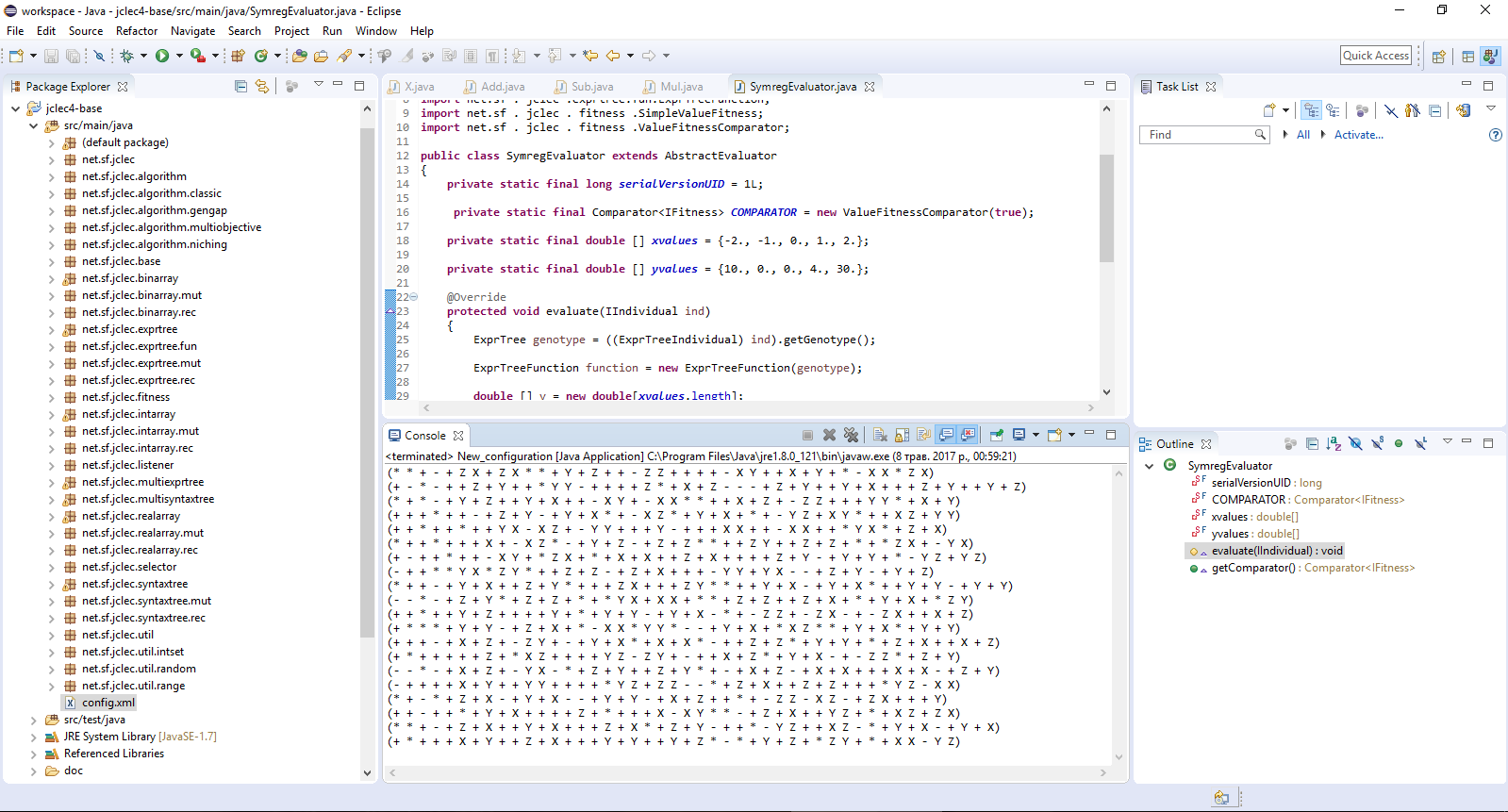
public Comparator<IFitness> getComparator()

{

return *COMPARATOR*;

}

}



**Рис 14.** Отримані результати

**Висновок**

Виконуючи дану розрахункову роботу, я ознайомилася із фреймворком **JCLEC** для роботи із еволюційними обчисленнями, навела короткі теоретичні відомості про його загальні характеристики, описала процес встановлення на прикладі середовищ Eclipse та NetBeans, навела кілька прикладів роботи та отримані у ході їх виконання результати.