ПРАКТИЧНЕ ЗАНЯТТЯ №9

Тема: Копіювання інформації з полів вводу у візуальних додатках Java (2 год.)

Мета: Навчитись копіювати інформацію з полів вводу та очищувати їх.

Обладнання: комп'ютери Pentium, Celeron.

Програмне забезпечення: MS Windows XP, пакет JDK.

Теоретичні відомості

Усі об'єкти колекцій можна комплексно обробляти. Наприклад, вивести на екран усі елементи, що містяться в списку. У разі масиву ми користуємося циклом for :

```
for (int i = 1; i < array.length; i++){
// // обробляємо елемент array[i] }
```

Маючи справу із списком, ми можемо поступити аналогічним чином, тільки замість array[i] писати array.get(i). Але ми не можемо поступити так з колекціями, елементи яких не індексуються (наприклад, чергою або множиною). А у разі індексованої колекції потрібно добре знати особливості її роботи: як визначити кількість елементів, як звернутися до елементу по індексу, чи може колекція бути розрідженою (тобто чи можуть існувати індекси, з якими не пов'язано ніяких елементів) і так далі

У програмуванні існує декілька випробуваних часом і прийомів структурної організації програми, що детально пропрацювали, званих патернами (шаблонами) проектування. Один з таких патернів називається Iterator. Ідея полягає в тому, що до колекції "прив'язується" об'єкт, єдине призначення якого - видати усі елементи цієї колекції в деякому порядку, не розкриваючи її внутрішню структуру.

У пакеті java.util описаний інтерфейс Iterator, що утілює цей патерн проектування. Він має всього три методи:

next() повертає черговий елемент колекції, до якої "прив'язаний" ітератор (і робить його поточним). Порядок перебору визначає сам ітератор.

hasNext() повертає true, якщо перебір елементів ще не закінчений

remove() видаляє поточний елемент

Інтерфейс Collection окрім розглянутих раніше методів, має метод iterator(), який повертає ітератор для цієї колекції, готовий до її обходу. За допомогою такого ітератора можна обробити усі елементи будь-якої колекції наступним простим способом:

```
Iterator iter = coll.iterator(); // coll - колекція, елементи якої ми хочемо обробити
while (iter.hasNext()){    //  обробляємо об'єкт, повертаний методом iter.next() }
```

Для колекцій, елементи яких проіндексовані, визначений більш функціональний ітератор, що дозволяє рухатися як в прямому, так і у зворотному напрямі, а також додавати в колекцію елементи. Такий ітератор має интрефейс ListIterator, успадкований від інтерфейсу Iterator і доповнюючий його наступними методами :

previous() - повертає попередній елемент (і робить його поточним);

hasPrevious() - повертає true, якщо попередній елемент існує (тобто поточний елемент не є першим елементом для цього ітератора);

add(Object item) - додає новий елемент перед поточним елементом;

set(Object item) - замінює поточний елемент;

nextIndex() і *previousIndex()* - служать для отримання індексів наступного і попереднього елементів відповідно.

У інтерфейсі List визначений метод listIterator(), повертаючий ітератор ListIterator для обходу цього списку.

Задача

Напишіть клас Student, що надає інформацію про ім'я студента методом *getName()* і про його курс методом *getCourse()*.

Напишіть метод *printStudents(List students, int course)*, який отримує список студентів і номер курсу і друкує в консоль імена тих студентів із списку, які навчаються на цьому курсі. Для обходу списку в цьому методі використовуйте ітератор.

Протестуйте ваш метод (для цього заздалегідь доведеться створити десяток об'єктів класу Student і помістити їх в список).