Содержание

ВВЕДЕНИЕ

1. Постановка задачи

2. Анализ задачи

3. Описание алгоритма

3.1 Алгоритм полного перебора

3.2 Генетический алгоритм перебора

4. Описание программы

4.1 Описание входных данных

4.2 Описание функций

4.3 Описание программы

5. Тестовые примеры

6. Анализ результатов работы

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ

Приложение А. Текст программы

**ВВЕДЕНИЕ**

В данной курсовой работе рассматривается задача нахождения оптимального расписания работы вычислительной машины. Расписание представляет собой порядок требований, поступающих на выполнение вычислительной машиной. Оптимальное расписание - это наилучший порядок требований, которые характеризуются определенными критериями.

Программа, реализующая задачу нахождения оптимального расписания, разработанная в данной курсовой работе, написана в среде Eclipse (язык программирования – Java). В программе реализованы 2 метода: метод полного перебора и генетический алгоритм перебора.

1. **Постановка задачи**

Пусть ЭВМ необходимо обслужить множество N = {1, 2, ..., n} требований. Требование k () поступает на обслуживание в момент времени dk >= 0 и для обслуживания требует tk единиц времени. Обслуживание требования k желательно завершить к директивному сроку Dk >= 0. Функция штрафа ϕk(x) = max(x - Dk, 0). Определить такой порядок выполнения программ, при котором суммарный штраф будет минимален. - фактическое время завершения обслуживания требования k.

**2. Анализ задачи**

Данная задача относится к классу задач упорядочивания.

Задано некоторое множество требований N = {1, 2, 3, …, n}, поступающих на обслуживание ЭВМ. Требуется составить такое расписание выполнения требований, при условии, что выполнены все требования и фактическое время выполнения каждого требования будет минимальным. Результат выполнения зависит от порядка поступления требований. При построении порядка учитываются такие параметры как: время поступления требования, время выполнения требования, директивный срок исполнения. Эти параметры влияют на конечное значение функции штрафа, которая вычисляется по формуле:



Таким образом, рассчитываем функцию штрафа для каждой перестановки, находим минимальный штраф и соответствующую последовательность.

Входные данные: время поступления требования dk (dk >= 0), время выполнения требования tk,директивный срок выполнения Dk (Dk >= 0).

Выходные данные: оптимальный порядок требований, суммарный штраф этого порядка, и время выполнения задачи алгоритмом полного перебора и генетическим алгоритмом перебора.

Исходные дан­ные будут вводиться в соответствующем поле графического интерфейса. Выходные данные – выводиться в поле результата.

**3. Описание алгоритма**

3.1 Алгоритм полного перебора

Полный перебор - метод решения математических задач. Сложность такого метода зависит от количества всех возможных решений задачи.

Метод подразумевает построение всех возможных расписаний, что в дальнейшем позволит называть этот метод точным. Для построенного множества расписаний высчитывается значение целевой функции для каждого элемента. В качестве результата возвращается расписание с лучшим значением целевой функции.

Алгоритм полного перебора, использованный в задаче:

1. Считывание данных и занесение их в соответствующие переменные
2. Формирование начального расписания
3. Вычисление факториала числа n (где n – количество требований)
4. Вычисление штрафа для текущего расписания и сравнение его с минимальным штрафом, если текущий штраф меньше минимального, то текущий штраф заносится в минимальный
5. Генерирование следующего расписания.

Алгоритм выполняется пока не будут рассмотрены все расписания, или пока текущее расписание не будет равным нулю. Число всех перестановок зависит от количества требований, поступивших на выполнение, и равно факториалу числа этих требований.

3.2 Генетический алгоритм перебора

Генетический алгоритм  — это [эвристический алгоритм](http://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих [биологическую](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) [эволюцию](http://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D1%8F). Является разновидностью [эволюционных вычислений](http://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D0%BE%D0%BB%D1%8E%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как [наследование](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [мутации](http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D1%82%D0%B0%D1%86%D0%B8%D1%8F), [отбор](http://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BE%D1%82%D0%B1%D0%BE%D1%80) и [кроссинговер](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%BD%D0%B3%D0%BE%D0%B2%D0%B5%D1%80).

Кроссинговер – обмен частями генетической цепи между лучшими представителями поколения.

Мутация – изменение случайного гена в цепи.

Для реализации генетического алгоритма используют следующие понятия: особь и популяция.

Применяя к нашей задаче: особь – это расписание требования, а популяция – это набор таких особей.

Алгоритм работы генетического алгоритма перебора, использованного в задаче:

1. Генерируем первую популяцию
2. Случайным образом выбираем 4 особи из текущей популяции и рассчитываем для них функцию штрафа (фитнесс функция)
3. Из этих 4 особей выбираем 3 особи с лучшей функцией штрафа, и применяем к ним мутирование
4. Для мутированных особей рассчитываем функцию штрафа и сравниваем со штрафом до мутации
5. Лучшие заносим в новую популяцию и повторяем пока не наберется новая популяция
6. Новую популяцию делаем текущей и повторяем алгоритм
7. Получаем популяцию лучших особей (с наименьшей функцией штрафа) и из них выбираем наименьшую.

Алгоритм полного перебора является более точным, но для решения использует больше времени и ресурсов, в отличии от генетического метода, который для решения задачи использует меньше времени и ресурсов, но дает неточный ответ при большом количестве требований.

**4. Описание программы**

4.1 Описание входных данных

Список переменных, используемых в программе:

* ***n*** – переменная типа int, количество требований
* ***dk[]*** – массив типа int, время поступления требований на выполнение
* ***tk[]*** – массив типа int, время выполнения требований
* ***Dk[]*** – массив типа int, директивные сроки выполнения
* ***best[]*** – массив типа int, лучшая последовательность
* ***min*** – переменная типа int, минимальный штраф
* ***t*** – переменная типа int, фактическое время завершения программы

4.2 Описание функций

Программа составления расписания представлена четырьмя классами: ***createGUI, Genetic, Perebor, Pack*.**

Класс ***createGUI*** – главный класс, связывает остальные классы программы, создание графического интерфейса пользователя.

Методы класса ***createGUI***:

* *createGUI()* – создание графического интерфейса пользователя
* *actionPerformed(ActionEvent e)* – занесение данных из полей графического интерфейса в соответствующие переменные
* *main()* – главный метод, запускает программу

Класс ***Genetic*** – реализует генетический алгоритм перебора. Методы класса ***Genetic***:

* *firstGen()* – генерация первой популяции
* *produceNextGen()* – создание новой популяции
* *mutate()* – метод мутации популяции
* *perform2()* – исполняющий метод, возвращает значение минимального штрафа и лучшей последовательности;
* *shtraf()* – метод для вычисления функции штрафа.

Класс ***Perebor*** – реализует алгоритм полного перебора. Методы класса ***Perebor***:

* *perform()* – исполняющий метод, возвращает значение минимального штрафа и лучшей последовательности
* *Next()* – генерирует следующую последовательность
* *Shtraf()* – метод для вычисления функции штрафа
* *fact() –* метод для вычисления факториала числа требований

Класс ***Pack*** – вспомогательный класс, служит для общения меду классами.

4.3 Описание программы

Данная программа представляет собой машинное решение поставленной задачи. Программа написана на объектно-ориентированном языке программирования – Java. Программа состоит из четырех классов.

Класс ***createGUI*** содержит метод main(), запускающий программа, а так же реализует графический интерфейс (Рис. 1).

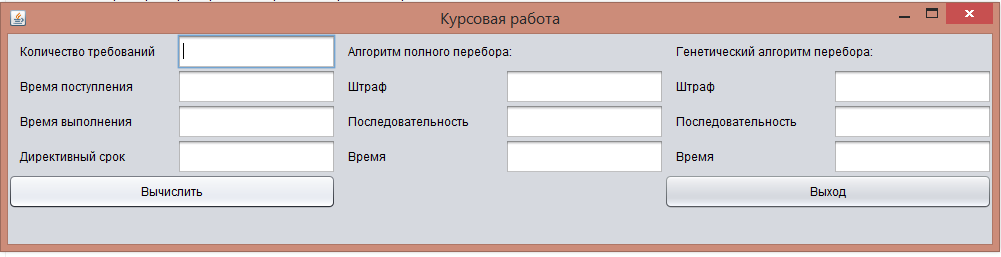


Рис.1 – Окно программы

В случае ввода неверных данных программа выдаст ошибку с сообщением о характере ошибки (Рис. 2).

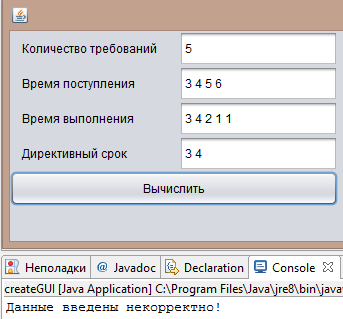


Рис. 2 – Результат работы программы при неверно введенных данных

**5. Тестовые примеры**

Пример 1:

Тестовый пример работы программы без штрафа:

Входные данные:

n: 3

dk: 0 1 2

tk: 2 1 3

Dk: 5 4 7

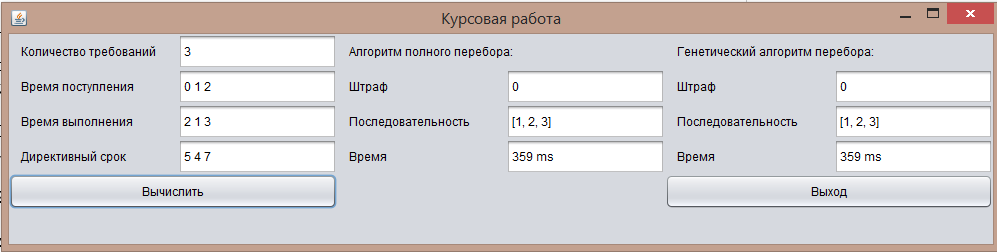


Рис. 3 – Результат выполнения примера 1.

Следовательно, оптимальная последовательность [1, 2, 3], т.к. штраф = 0, является минимальным.

Пример 2:

Тестовый пример работы программы со штрафом:

Входные данные:

n: 3

dk: 0 5 3

tk: 2 4 5

Dk: 6 4 3

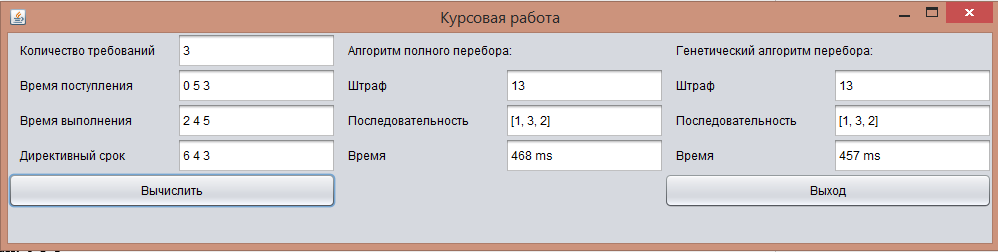


Рис. 4 – Результат выполнения примера 2

Следовательно, оптимальная последовательность [1, 3, 2], т.к. штраф=13, является минимальным.

Пример 3:

Тестовый пример работы программы со штрафом показывающий разницу во времени исполнения алгоритмов:

Входные данные:

n: 8

dk: 0 8 4 2 3 6 87 4

tk: 10 12 3 14 20 8 9 23

Dk: 10 53 23 36 54 23 2 3

По данным выполнения программы видим, что выполнение алгоритма полного перебора занимает больше времени, чем генетический алгоритм перебора.

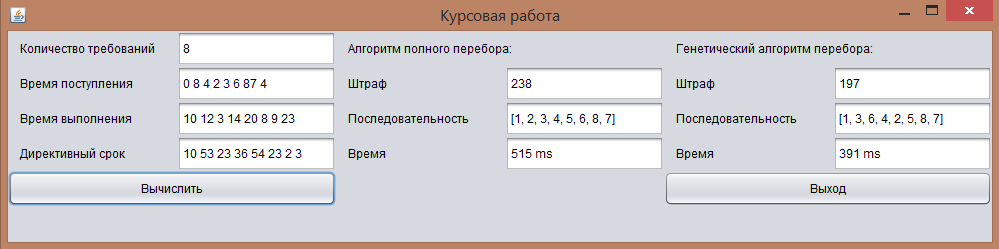


Рис. 5 – Результат выполнения примера 3

Результаты выполнения программы на тестовых примерах совпали с ожидаемыми, из чего делаем заключение о правильности работы программы.

**6. Анализ результатов работы**

В результате работы программы были получены значения таких величин, как оптимальное расписание выполнения программ, соответствующая ему функция штрафа, время выполнения.

Метод полного перебора даёт однозначно точное решение поставленной задачи без каких-либо погрешностей и неточностей. Однако это достигается путём сложных и многочисленных вычислений, которые для реализации требуют значительных ресурсов.

Таким образом, метод полного перебора можно применять при сравнительно небольших данных, строгой желательности точного ответа и относительно небольших сроках решения задачи. Число программ не должно не превышать 10.

Генетический алгоритм, при небольшом количестве программ, уступает в эффективности методу полного перебора. алгоритм даёт приближенное к точному значение результата с погрешностью 10-15%. В порядке случаев алгоритм может вернуть приближенное значение с точностью до 5%, а в некоторых случаях - и само точное решение. Это объясняется тем, что изначальный генотип выбирается случайным образом.

По экспериментально полученным данным построим графики зависимости времени от количества требований для алгоритма полного перебора и генетического алгоритма, в диапазоне от 3 до 11 требований.

Рис. 6 – График зависимости времени от количества требований.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была разработана программа, которая выполняет поставленную задачу: построение порядка обслуживания требований, при котором суммарный штраф их обработки будет минимальным.

Программа находит наилучшую последовательность выполнения этих требований и расчет значений штрафа двумя реализованными алгоритмами: алгоритмом полного перебора и генетическим алгоритмом перебора.

**СПИСОК ЛИТЕРАТУРЫ**

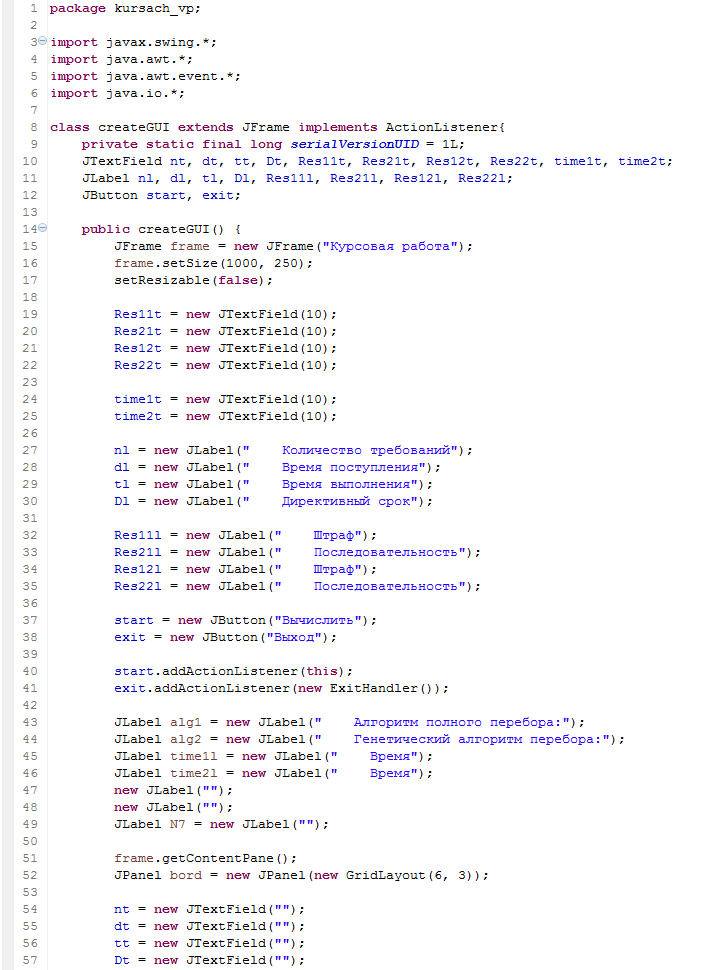
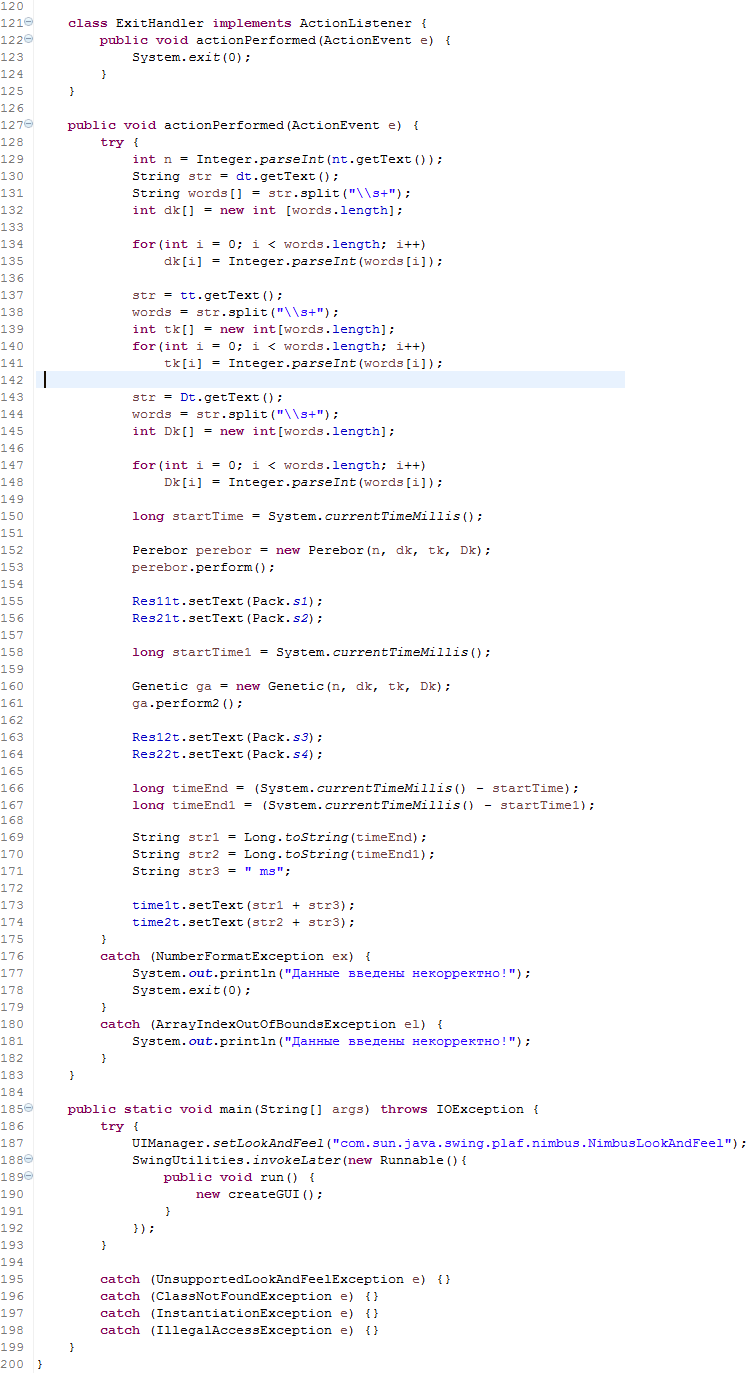
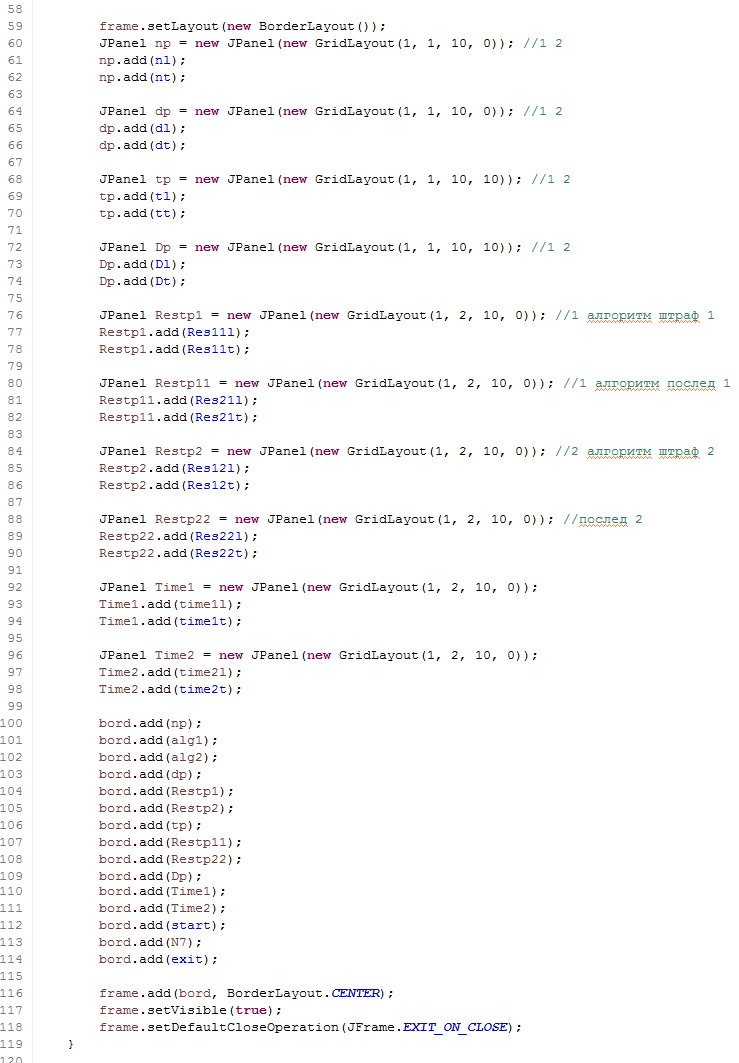
1. Методические указания к выполнению курсовой работы по дисциплине «Вычислительный практикум» для студентов заочной формы обучения специальности 6.05010201, Сост. А.В. Скатков, Г.Г. Сергеев, Л.П. Луговская. – Севастополь: Изд-во СевНТУ, 2012. – 23 с.

2. Ноутон. П. JAVA 2/ П.Ноутон, Г. Шилдт.– Спб:ВНУ, 2007.–1034 с.

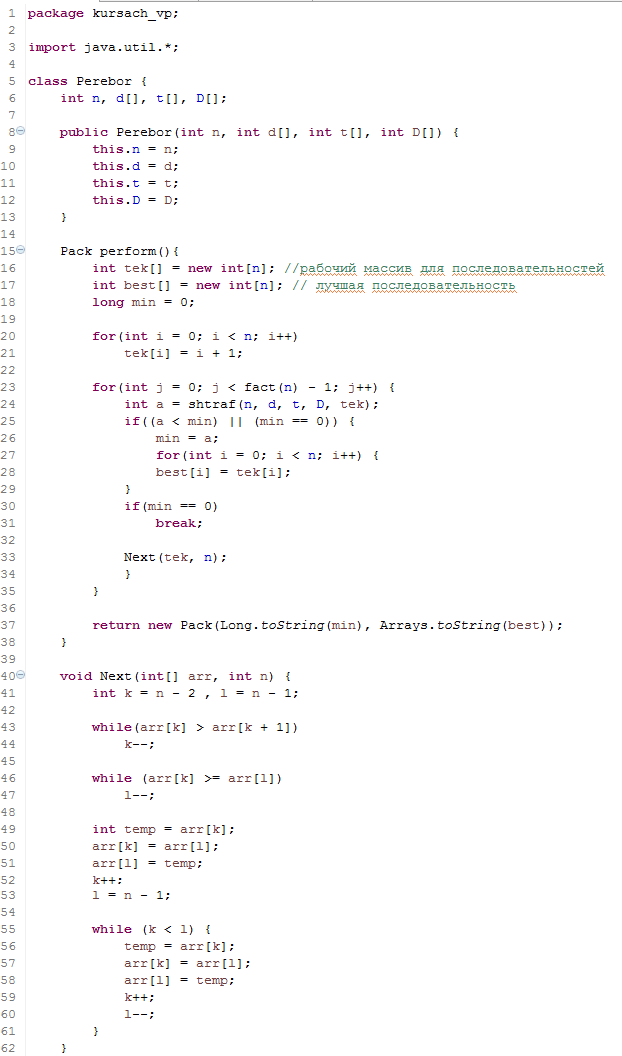
3.Портянкин И. Swing Эффектные пользовательские интерфейсы/ И.Портянкин.– Спб: ПИТЕР,2005.–528 с.

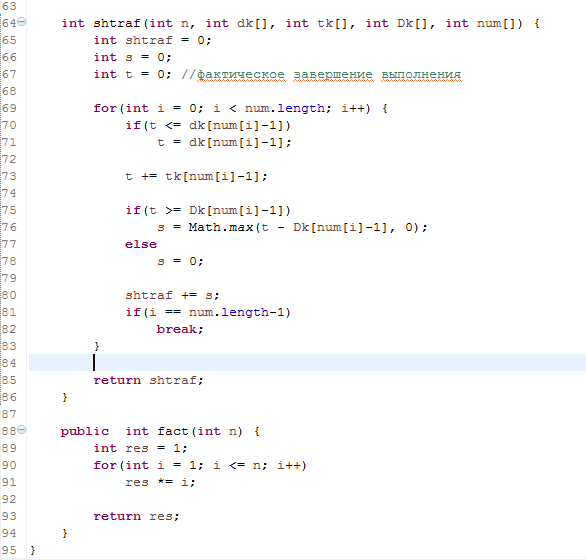
**Приложение А. Текст программы**

Класс createGUI

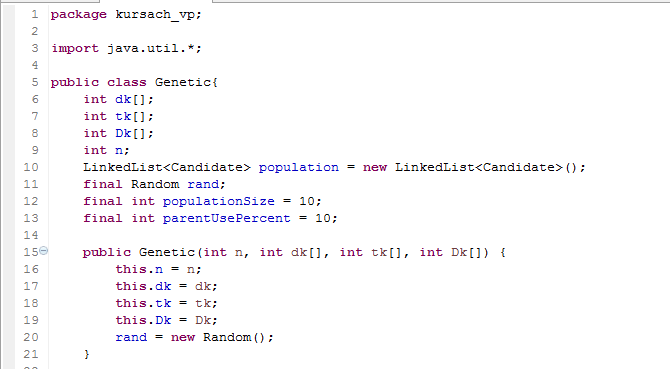
 

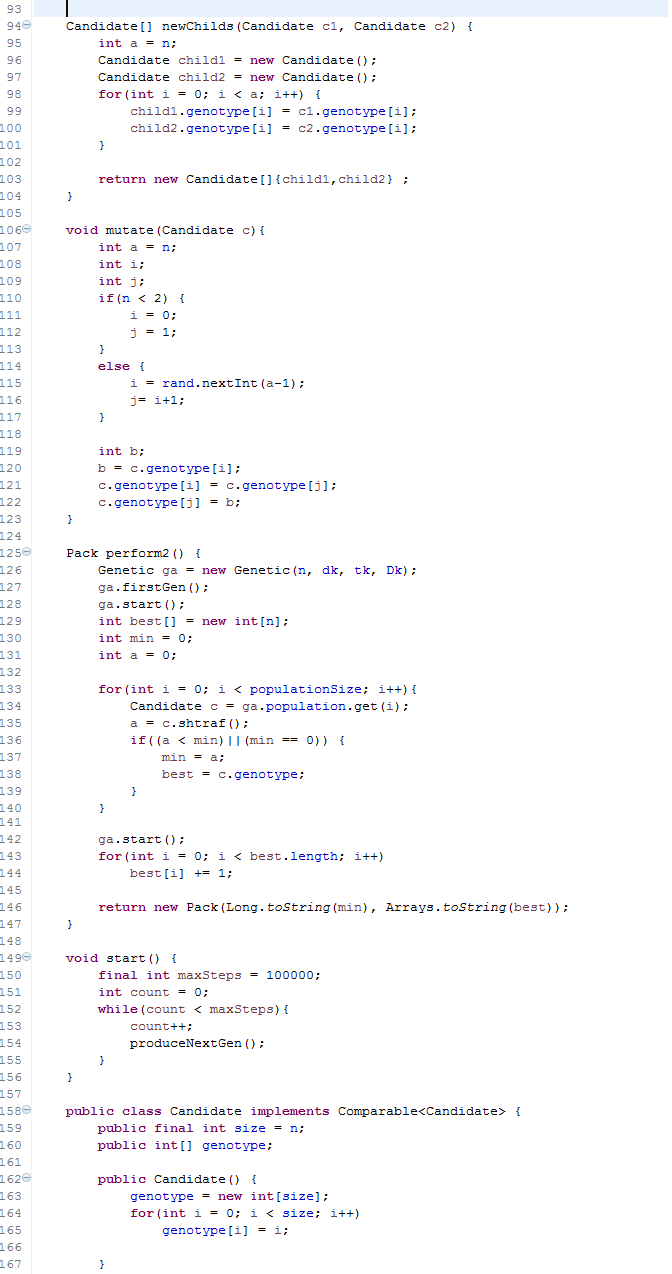
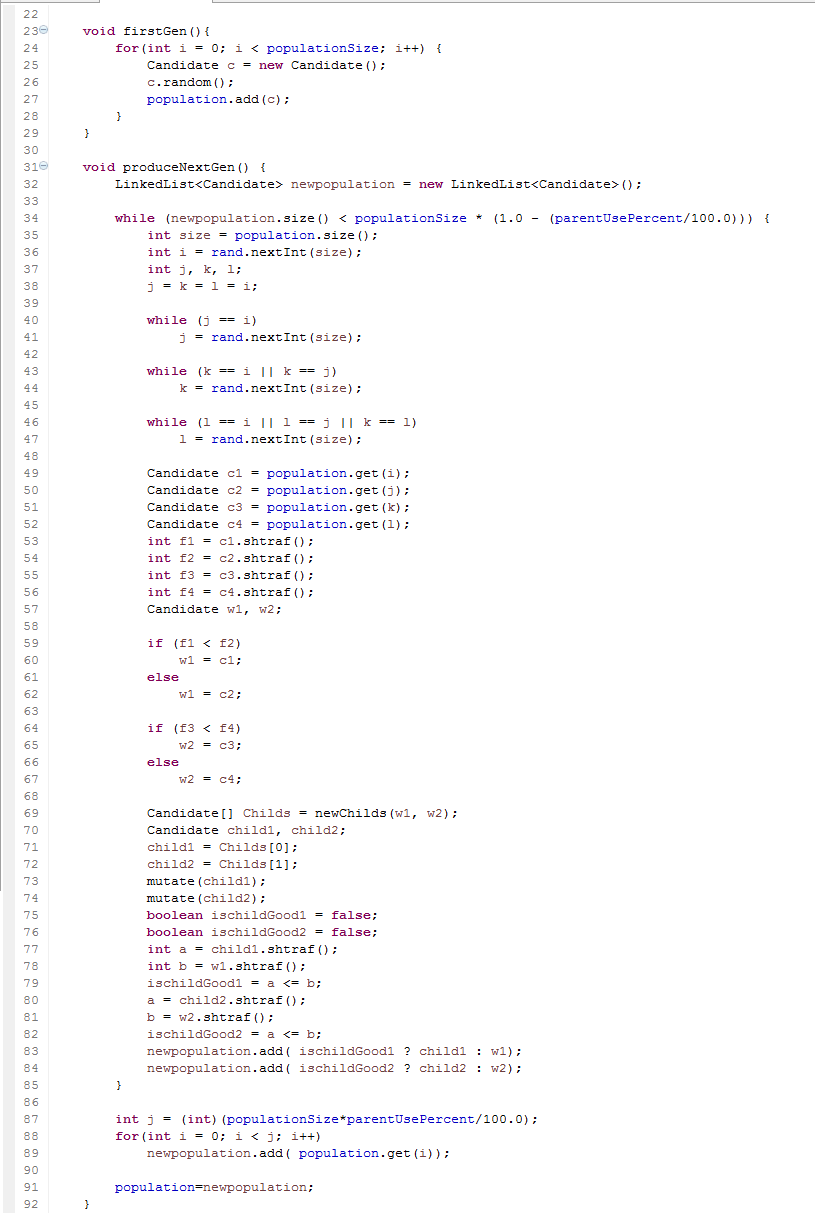
Класс Perebor:





Класс Genetic





Класс Pack:

