Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем

и технологий»

**Отчёт**

по лабораторной работе №04

по дисциплине ***«Объектно-ориентированные технологии***

***программирования и стандарты проектирования»***

тема: «Условные конструкции. Разветвляющиеся алгоритмы»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701119  Прадед Кирилл Витальевич |
| Преподаватель: |  | Иванченко Виктор Викторович |

2020-2021 учебный год

ЛАБОРАТОРНАЯ РАБОТА #04

Базовый синтаксис языка Java. Условные конструкции. Разветвляющиеся алгоритмы

**Цель работы**

Изучить синтаксис условной конструкции и оператора множественного выбора языка Java для реализаций разветвляющихся алгоритмов (ветвлений) и закрепить их на примере разработки простейших интерактивных консольных Java-приложений.

**Tребования**

1) Необходимо выполнить весь блок основных заданий, по одному соответствующему заданию – из индивидуальных заданий А и В. Дополнительное задание выполняется на Ваше усмотрение.

2) Для каждого вычислительного алгоритма необходимо спроектировать блоксхему решения, которую необходимо поместить в отчёт.

3) На базе спроектированных алгоритмов разработать простейшее консольное интерактивное приложение с использование архитектурного шаблона проектирования Model-View-Controller, MVC.

4) Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: by.bntu.fitr.poisit.nameofstudent.javalabs.lab04.

5) При выполнения задания необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый, легко поддерживаемый и читаемый код.

6) Также рекомендуется придерживаться Single Responsibility Principle, SRP (принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.

7) Если логически не подразумевается или в задании иного не указано, то входными и выходными данными являются вещественные числа (числа с плавающей запятой).

8) Все задания необходимо решать используя только базовые операции (простые операторы), определённые над примитивными типами данных в языке программирования Java, и условные конструкции (т.е. не нужно использовать циклические конструкции, массивы, строковые данные и операции над ними и т.д.).

9) В соответствующих компонентах бизнес-логики необходимо предусмотреть «защиту от дурака».

10)Для генерирования случайных чисел воспользуйтесь методами объекта класса java.util.Random, а для реализации ввода данных с консоли (терминала) – соответствующими методами объекта класса java.util.Scanner. 11)Программа должна обязательно быть снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, название бригады (если есть), номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.

12)Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем. Интерфейс программы и комментарии в коде должны быть на английском языке.

13) При проверки работоспособности приложения необходимо проверить все тестовые случаи. 14)При выполнении задания не рекомендуется использовать интегрированные средства разработки (Integrated Development Environment, IDE). Следует задействовать любой текстовый редактор и основные компоненты JDK (компилятор – javac, утилиту для запуска JVM – java).

15)При разработке программ придерживайтесь соглашений по написанию кода на Java (Java Code-Convention)

**Основное задание**

1) В молодом возрасте дракон каждый год отращивает по три головы, но после того, как ему исполнится 200 лет – только по две, а после 300 лет – лишь по одной. Разработайте программу, которая высчитывала бы, сколько голов и глаз у дракона, которому N лет. Считать, что при рождении у дракона имеется уже три головы.

2) Напишите программу «The Greatest», которая определяет какое из четырёх (или пяти, или шести и т.д.) введённых пользователем значений наибольшее (наименьшее). Предусмотреть возможность равенства всех значений.

3) Напишите программу, которая определяла бы, является ли заданное число кратным соответствующим числам 2, 3, 5, 7, 11, 13, 17 и 19.

Результаты выполнения задания:

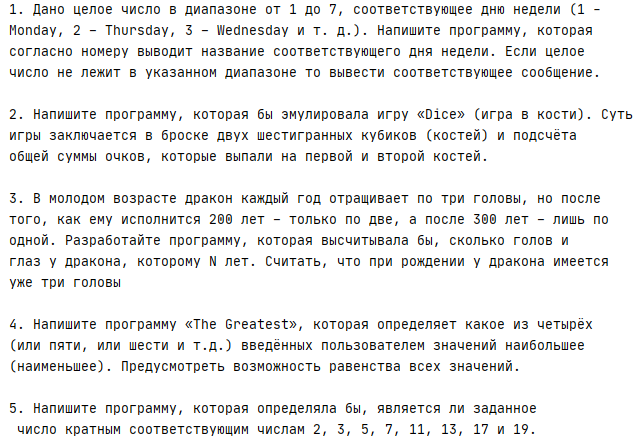


Рисунок 1. Главное меню консольного приложения.

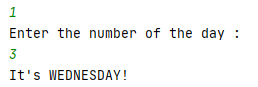


Рисунок 2. Результат выполнения первого пункта

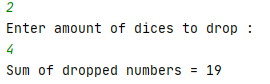


Рисунок 3. Результат выполнения второго пункта

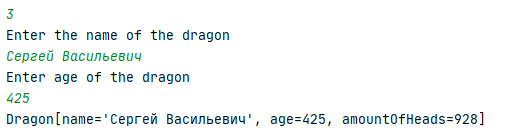


Рисунок 4. Результат выполнения третьего пункта.

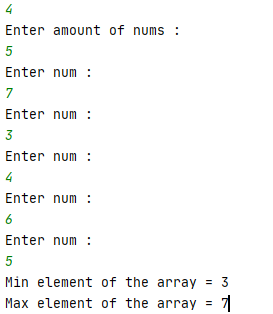
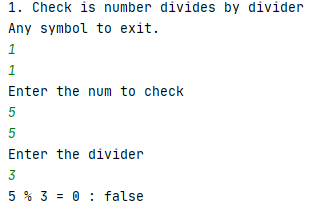


Рисунок 5. Результат выполнения четвертого пункта.



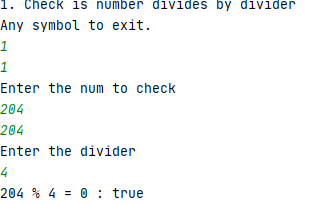


Рисунок 6-7. Результат выполнения пятого пункта.

Что мы узнали нового в процессе выполнения лабораторной работы:

1. В Java весь код пишется в пользовательских типах данных, таких как класс, интерфейс или перечисление.

2. Имя класса всегда пишется с БОЛЬШОЙ буквы и, как правило, обозначает соответствующее существительное в единственном числе!

3. Чтобы в процессе выполнения программы можно было хранить начальные, промежуточные и результирующие значения (данные) в языках программирования используются переменные.

4. В общем случае, переменная (variable) – поименованная область памяти, которую можно использовать для хранения данных и осуществления доступа к ним. Данные, которые находятся в переменной (т.е. по данному адресу памяти), называют значением (value) данной переменной.

5. Переменные упрощают написание кода программы, делают этот код читабельным и легко поддерживаемым.

6. Объявление переменной – это определение её типа и имени.

*Приложение А. Код программы.*

**Задание Day of Week**

**package** by.praded.dayofweek.controller;  
  
**public class** DayController {  
  
 **public static void** start() {  
 **try** {  
 DayView.*printDay*(DayService.*getDayByIndex*(DayService.*registerDay*(DayView.*enterDay*())));  
 } **catch** (NoSuchDayException e) {  
 e.printStackTrace();  
 }  
 }  
}

**package** by.praded.dayofweek.exception;  
  
**public class** NoSuchDayException **extends** Exception {  
 **public** NoSuchDayException(String message) {  
 **super**(message);  
 }  
}

**package** by.praded.dayofweek.model;  
  
**public class** Day {  
 **private int number**;  
  
 **public** Day(**int** number) {  
 **this**.**number** = number;  
 }  
  
 **public int** getNumber() {  
 **return number**;  
 }  
  
 **public void** setNumber(**int** number) {  
 **this**.**number** = number;  
 }  
}

**package** by.praded.dayofweek.service;  
  
  
**public class** DayService {  
 **public static** Day registerDay(**int** number) **throws** NoSuchDayException {  
 **if** (number < 1|| number>7){  
 **throw new** NoSuchDayException(**"Number of the day is invalid"**);  
 }  
 **return new** Day(number);  
 }  
 **public static** String getDayByIndex(Day day){  
 **return** DayOfWeek.*of*(day.getNumber()).name();  
 }  
}

**package** by.praded.dayofweek.view;  
  
**public class** DayView {  
 **private static final** Scanner ***SCANNER*** = **new** Scanner(System.***in***);  
  
 **public static int** enterDay(){  
 System.***out***.println(**"Enter the number of the day : "**);  
 **return *SCANNER***.nextInt();  
 }  
  
 **public static void** printDay(String day) {  
 System.***out***.println(**"It's "** + day + **"!"**);  
 }  
}

**Задание Dice.**

**package** by.praded.dice.controller;  
  
**public class** DiceController {  
 **public static void** start() {  
 Dice dice = **new** Dice();  
 DiceView.*printResult*(DiceService.*dropDice*(dice, DiceView.*enterAmountOfDices*()));  
  
 }  
}

**public class** Dice {  
 **private final int**[] **values**;  
  
 **public** Dice() {  
 **this**.**values** = **new int**[]{1, 2, 3, 4, 5, 6};  
 }  
  
 **public int** size(){  
 **return values**.**length**;  
 }  
  
 **public int** findElement(**int** idx) {  
 **return values**[idx];  
 }  
}

**package** by.praded.dice.service;  
  
**public class** DiceService {  
 **private static final** Random ***random*** = **new** Random();  
  
 **public static int** dropDice(Dice dice, **int** amountOfDrops) {  
 **int** counter = 0;  
 **for**(**int** i = 0; i < amountOfDrops; i++) {  
 counter += dice.findElement(***random***.nextInt(6));  
 }  
 **return** counter;  
 }  
}

**package** by.praded.dice.view;  
  
**public class** DiceView {  
 **private static final** Scanner ***SCANNER*** = **new** Scanner(System.***in***);  
  
 **public static int** enterAmountOfDices() {  
 System.***out***.println(**"Enter amount of dices to drop : "**);  
 **return *SCANNER***.nextInt();  
 }  
  
 **public static void** printResult(**int** result) {  
 System.***out***.println(**"Sum of dropped numbers = "** + result);  
 }  
}

**Задание Dragon**

**package** by.praded.dragon.controller;  
  
**public class** DragonController {  
 **public static void** start() {  
 Dragon dragon = **new** Dragon(DragonView.*enterDragonName*(), DragonView.*enterDragonAge*());  
 DragonView.*printDragonInfo*(dragon);  
 }  
}

**package** by.praded.dragon.model;  
  
**import** by.praded.dragon.service.DragonService;  
  
*/\*\*  
 \** ***@author*** *Kiryl Praded  
 \* 28.10.2020  
 \*/***public class** Dragon {  
 **private** String **name**;  
 **private int age**;  
 **private int amountOfHeads**;  
  
 **public** Dragon(String name, **int** age) {  
 **this**.**age** = age;  
 **this**.**name** = name;  
 **amountOfHeads** = DragonService.*countHeads*(**this**);  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public int** getAge() {  
 **return age**;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.**age** = age;  
 **amountOfHeads** = DragonService.*countHeads*(**this**);  
 }  
  
 **public int** getAmountOfHeads() {  
 **return amountOfHeads**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Dragon["** +  
 **"name='"** + **name** + **'\''** +  
 **", age="** + **age** +  
 **", amountOfHeads="** + **amountOfHeads** +  
 **']'**;  
 }

**package** by.praded.dragon.service;  
  
**public class** DragonService {  
 **private static** DragonService *dragonService*;  
  
 **private** DragonService(){  
  
 }  
  
 **public static** DragonService getDragonService() {  
 **if** (*dragonService* == **null**){  
 *dragonService* = **new** DragonService();  
 }  
 **return** *dragonService*;  
 }  
  
 **public static int** countHeads(Dragon dragon) {  
 **int** age = dragon.getAge();  
 **int** amountOfHeads = 3;  
 **for** (**int** i = 0; i < age; i++) {  
 **if** (i < 200) {  
 amountOfHeads += 3;  
 } **else if** (i < 300) {  
 amountOfHeads += 2;  
 } **else** {  
 amountOfHeads +=1;  
 }  
 }  
 **return** amountOfHeads;  
 }  
}

**package** by.praded.dragon.view;  
  
**public class** DragonView {  
 **private static final** Scanner ***SCANNER*** = **new** Scanner(System.***in***);  
  
 **public static** String enterDragonName() {  
 System.***out***.println(**"Enter the name of the dragon"**);  
 **return *SCANNER***.nextLine();  
 }  
  
 **public static int** enterDragonAge() {  
 System.***out***.println(**"Enter age of the dragon"**);  
 **return *SCANNER***.nextInt();  
 }  
  
 **public static void** printDragonInfo(Dragon dragon) {  
 System.***out***.println(dragon);  
 }  
}

**Задание The Greatest**

**package** by.praded.greatest.controller;  
  
  
  
**public class** GreatestController {  
 **public static void** start() {  
 **int**[] nums = GreatestView.*enterNums*();  
 GreatestView.*printMaxAndMin*(GreatestService.*findMin*(nums), GreatestService.*findMax*(nums));  
 }  
  
}

**package** by.praded.greatest.service;  
  
**public class** GreatestService {  
  
 **public static int** findMin(**int**[] arr) {  
 **int** min = arr[0];  
 **for** (**int** i : arr) {  
 **if** (i < min) {  
 min = i;  
 }  
 }  
 **return** min;  
 }  
  
 **public static int** findMax(**int**[] arr) {  
 **int** max = arr[0];  
 **for** (**int** i : arr) {  
 **if** (i > max) {  
 max = i;  
 }  
 }  
 **return** max;  
 }  
}

**package** by.praded.greatest.view;  
  
**public class** GreatestView {  
 **public static int**[] enterNums(){  
 Scanner scanner = **new** Scanner(System.in);  
 System.out.println(**"Enter amount of nums : "**);  
 **int** amount = scanner.nextInt();  
 **int**[] nums = **new int**[amount];  
 **for**(**int** i = 0; i < amount; i++) {  
 System.out.println(**"Enter num : "**);  
 nums[i] = scanner.nextInt();  
 }  
 **return** nums;  
 }  
  
 **public static void** printMaxAndMin(**int** min, **int** max) {  
 System.out.println(**"Min element of the array = "** + min  
 + **"\nMax element of the array = "** + max);  
 }  
}

**Задание Multiple**

**package** by.praded.multiple.controller;  
  
**public class** MultipleController {  
 **public static void** start() {  
 **boolean** start = **true**;  
 **while** (start){  
 String command = MultipleView.*enterCommand*();  
 **switch** (command) {  
 **case "1"**:  
 **int** toCheck = MultipleView.*enterNum*(**"Enter the num to check"**);  
 **int** divider = MultipleView.*enterNum*(**"Enter the divider"**);  
 MultipleView.*printResult*(toCheck, divider, MultipleService.*isMultiple*(toCheck, divider));  
 **break**;  
 **default**:  
 start = **false**;  
 }  
 }  
 }  
}

**package** by.praded.multiple.service;  
  
**public class** MultipleService {  
 **public static boolean** isMultiple(**int** toCheck, **int** check){  
 **return** toCheck % check == 0;  
 }  
  
}

**package** by.praded.multiple.view;  
  
**public class** MultipleView {  
 **private static final** Scanner ***SCANNER*** = **new** Scanner(System.***in***);  
  
 **public static** String enterCommand() {  
 System.***out***.println(**"1. Check is number divides by divider"** + **"\nAny symbol to exit."**);  
 ***SCANNER***.nextLine();  
 **return *SCANNER***.nextLine();  
 }  
  
 **public static void** printResult(**int** toCheck, **int** check, **boolean** result) {  
 System.***out***.println(toCheck + **" % "** + check + **" = 0 : "** + result);  
 }  
  
 **public static int** enterNum(String str){  
 System.***out***.println(str);  
 ***SCANNER***.nextLine();  
 **return *SCANNER***.nextInt();  
 }  
}

Контрольные вопросы

1) Для чего в языках программирования нужны управляющие конструкции (операторы)? Чем они управляют?

Управляют процессом выполнения программы. Необходимы для решения тех или иных логических задач на основе результата которых решается дальнейшее выполнение программы.

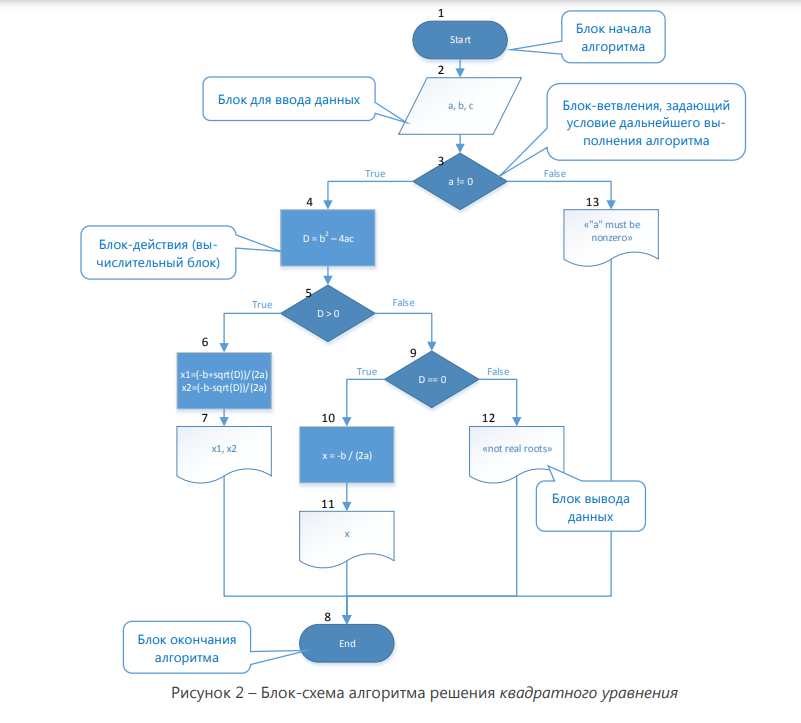
2) Какие существуют в языках программирования фундаментальные типы управляющих конструкций, которые используются при построении любой компьютерной программы? Какие алгоритмы они покрывают?

1-2-многоветвевые.

3) Почему необходимо в начале проектировать систему (к примеру, разрабатывать алгоритмы решения задачи), а затем её реализовывать (преступать к непосредственному написанию кода программы)?

Потому что так проще структурировать код и определить как он должен работать. После этого остается лишь одна сложная задача - реализовать логику кода.

4) Опишите основные (базовые) элементы блок-схемы для графического предоставления алгоритмов решения задач?



5) Какие алгоритмы называются разветвляющими и где они применяются?

Условные конструкции с несколькими вариантами развития событий

6) Какие в языке Java существуют разновидности условных конструкций?

If-else if- else switch

7) Как в Java реализуется механизм истинности-ложности?

8) Могут ли различные значения (объекты) выступать в качестве условия (или условного выражения) истинности-ложности в Java? Какое при этом используется правило?

Да могут. Это должны быть Boolean значения

9) Синтаксис универсальной условной конструкции if-else и её обозначение в виде блок-схемы? Как она работает? Когда данная конструкция применятся?

П.4.

If (Boolean expression) {

}else {

}

10)Какой логике обычно следуют разработчики при размещении соответствующих инструкций (последовательности операторов) для выполнения после ключевых слов if и else?

11)Какую разновидность условной конструкции if-else необходимо использовать, если необходимо выполнить только одно действие при определённом условии (истинности или ложности условия)? Как выглядит она с помощью блоксхемы? Какое используется условие, чтобы данная конструкция была в коде более читабельна?

If() {

}

12)Какой в Java используется подход для реализации выполнения кода, в котором задаётся больше двух условий выполнения? Как выглядит данный подход с помощью блок-схемы?

Используются различные операторы && || |

13)Какие есть способы форматирования вышеописанного подхода? Какой из этих способов лучше использовать согласно соглашению?

Автоформатирование,форматирование вручную, форматирование через некоторые плагины.

14)Каков синтаксис оператора множественного выбора switch-case-default и его обозначение в виде блок-схемы? Как он работает?

Как условная конструкция. в процессе выполнения switch станет условной конструкцией

15)Как происходит каскадное выполнение case-ов в операторе множественного выбора switch?

Без оператора прерывания кейсы выполняются один за одним.

16)Зачем нужна группировка case-ов в условном операторе множественного выбора?

17)Для чего нужно в условном операторе множественного выбора ключевое слово default и особенность его использования?

Это как конструкция else, будет выполнятся если не один из предыдущих кейсов не подошел.

18)Какие типы данных можно использовать для переменной в конструкции switch-case-default? Что поменялось в синтаксисе конструкции с выходом различных версий JDK?

boolean

19)Чем условная конструкция множественного выбора switch-case-default будет отличаться от универсальной классической условной конструкции if-else? Когда они применяются в коде?

Является не более чем синтаксическим сахаром.

20)В каких случаях употребление условной (или «тернарной») операции будет лучше, чем условной конструкции if-else, а в каком случае наоборот?

Очень удобно использовать тернарный оператор в return или при присваивании.