НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Автоматики и Вычислительной Техники

Кафедра Вычислительной Техники

**ЛАБОРАТОРНАЯ РАБОТА № 5**

***Потоки ввода-вывода.***

***Сериализация объектов в файл***

**Группа:** АВТ-710

**Студент:** Шелудько В.

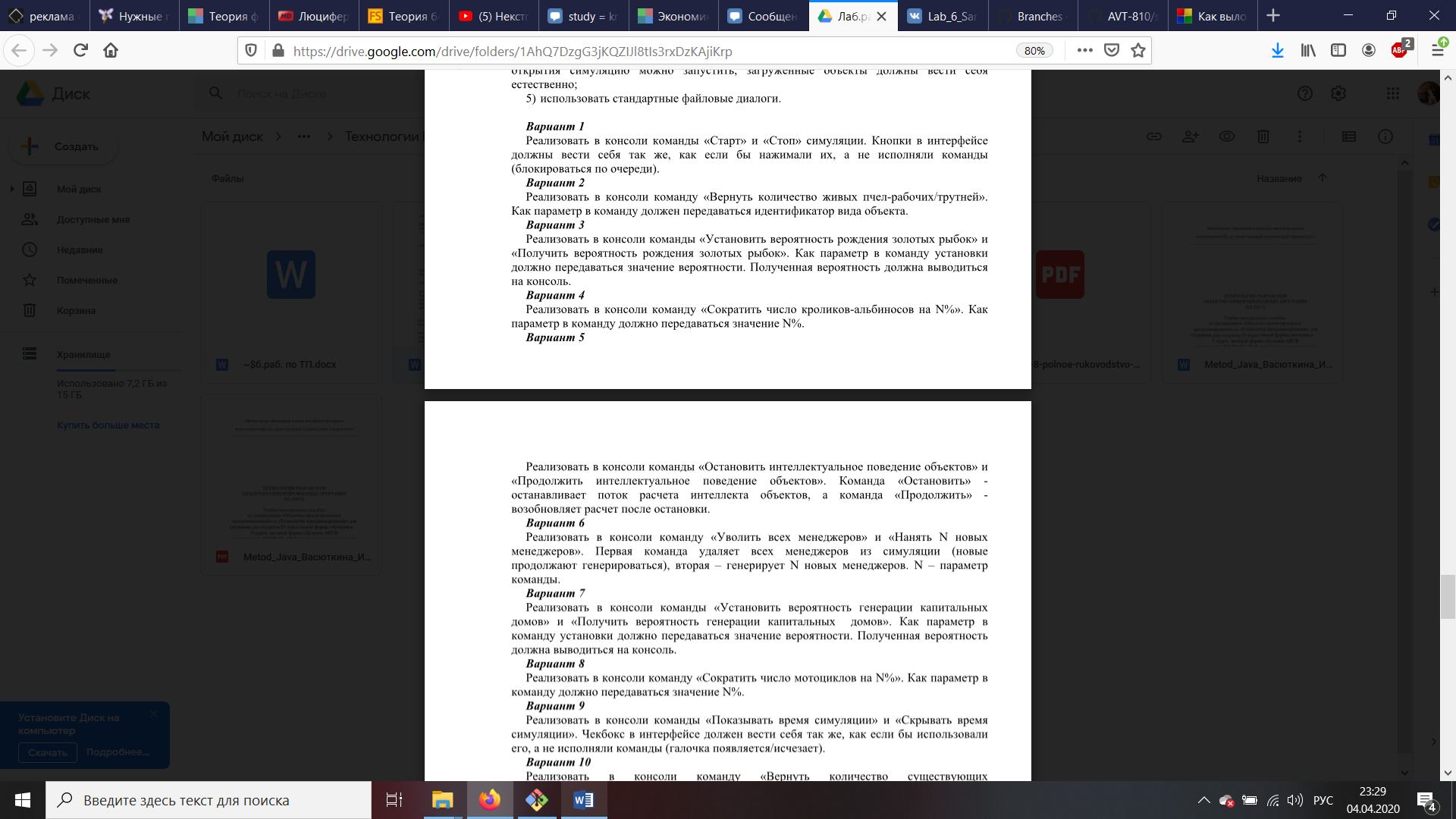
Бурдинский И.

**Вариант:** №6

**Преподаватель:** Михайленко Д.А.

**Цели работы:**

1. Изучить особенности реализации системы ввода-вывода в Java.
2. Доработать программу, созданную в лабораторной работе № 4:
3. добавить в главное меню команду «Консоль». По этой команде должно появляться немодальное диалоговое окно с многострочным текстовым полем, занимающим всю область окна. В это окно можно вводить команды по варианту. В это же окно выводится реакция программы на команду;
4. для передачи команды в основное окно программы использовать каналы ввода-вывода;
5. создать конфигурационный файл для программы. В конфигурационный файл должны сохраняться все настройки симуляции, т.е. все данные и состояния, которые задаются в панели управления программы. Конфигурационный файл должен читаться при запуске программы и записываться при выходе. Формат файла текстовый;
6. добавить в главное меню пункты «Загрузить» и «Сохранить». Команда «Сохранить» вызывает сериализацию всех «живых» объектов в ней. Команда «Загрузить» останавливает текущую симуляцию (если симуляция запущена) и загружает объекты из выбранного файла. Не забудьте скорректировать время рождения объектов. После открытия симуляцию можно запустить, загруженные объекты должны вести себя естественно;
7. использовать стандартные файловые диалоги.



**Постановка задачи:**

Написать класс-поток Консоли нашего приложения. Консоль должна поддерживать ввод команд и их отправку в основной поток приложения. Вывод информации о введенной команде

Добавить кнопки сохранения\загрузки коллекции объектов в файл, при загрузке объектов из файла, коллекция полностью обнуляется.

Сохранение\загрузка параметров симуляции при закрытии\запуске приложения

**Описание функционала:**

Сохранение и загрузка параметров симуляции в\из текстового файла

void saveConfig() throws Exception{  
 BufferedWriter out = new BufferedWriter(new FileWriter("Config.txt"));  
  
 out.write(String.*valueOf*((Worker.*Chance*)));  
 out.write('\n');  
 out.write(String.*valueOf*((Worker.*TimeBetweenSpawn*)));  
 out.write('\n');  
 out.write(String.*valueOf*((Worker.*LifeTime*)));  
 out.write('\n');  
 out.write(String.*valueOf*((Manager.*Chance*)));  
 out.write('\n');  
 out.write(String.*valueOf*((Manager.*TimeBetweenSpawn*)));  
 out.write('\n');  
 out.write(String.*valueOf*((Manager.*LifeTime*)));  
  
 out.close();  
}  
  
void loadConfig() throws Exception{  
 BufferedReader in = new BufferedReader(new FileReader("Config.txt"));  
  
 Worker.*Chance* = Float.*parseFloat*(in.readLine());  
 Worker.*TimeBetweenSpawn* = Float.*parseFloat*(in.readLine());  
 Worker.*LifeTime* = Float.*parseFloat*(in.readLine());  
 Manager.*Chance* = Float.*parseFloat*(in.readLine());  
 Manager.*TimeBetweenSpawn* = Float.*parseFloat*(in.readLine());  
 Manager.*LifeTime* = Float.*parseFloat*(in.readLine());  
  
 txtFieldSpawnTimeWorker.setText(String.*valueOf*(Worker.*TimeBetweenSpawn*));  
 txtFieldSpawnTimeManager.setText(String.*valueOf*(Manager.*TimeBetweenSpawn*));  
  
 txtFieldLifeTimeWorker.setText(String.*valueOf*(Worker.*LifeTime*));  
 txtFieldLifeTimeManager.setText(String.*valueOf*(Manager.*LifeTime*));  
  
 sliderChanceWorker.setValue( (int)(Worker.*Chance*\*100));  
 sliderChanceManager.setValue((int)(Manager.*Chance*\*100));  
  
 in.close();  
}

Сохранение и загрузка объектов “поштучно” средствами сериализации java, при загрузке информация о текущих объектах обнуляется и загружается с файла. Также, корректируется время рождения

void saveObject() throws IOException {  
 synchronized (\_habitat.ObjCollection){  
 ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("data.dat"));  
 oos.writeObject(\_habitat.ObjCollection);  
 oos.close();  
 }  
}

void loadObject() throws IOException, ClassNotFoundException {  
 synchronized (\_habitat.ObjCollection){  
 FileInputStream f = new FileInputStream("data.dat");  
 ObjectInputStream ois = new ObjectInputStream(f);  
  
 \_habitat.ObjCollection.clear();  
 \_habitat.IdCollection.clear();  
 \_habitat.BornTimeCollection.clear();  
  
 Employee.*Amount* = 0;  
 Worker.*Amount* = 0;  
 Manager.*Amount* =0;  
  
 LinkedList<Employee> tmp = null;  
  
 if(f.available() > 0)  
 tmp = (LinkedList<Employee>) ois.readObject();  
 else  
 return;  
 ois.close();  
  
 for(Employee emp : tmp)  
 {  
 emp.setBornTime(*ElapsedTime*);  
 Employee.*Amount*++;  
 \_habitat.IdCollection.add(emp.getID());  
 \_habitat.BornTimeCollection.put(emp.getID(), emp.getBornTime());  
 if(emp instanceof Worker)  
 Worker.*Amount*++;  
 else  
 Manager.*Amount*++;  
 }  
 \_habitat.ObjCollection = tmp;  
 }  
}

Поиск команды в введенной пользователем строки

tern Patre1 = Pattern.*compile*("^ \*?dismiss$",Pattern.*MULTILINE*);  
Pattern re2 = Pattern.*compile*("^ \*?hire ([0-9]{1,3})$",Pattern.*MULTILINE*);

Содержимое файла конфигурации

0.79

10.0

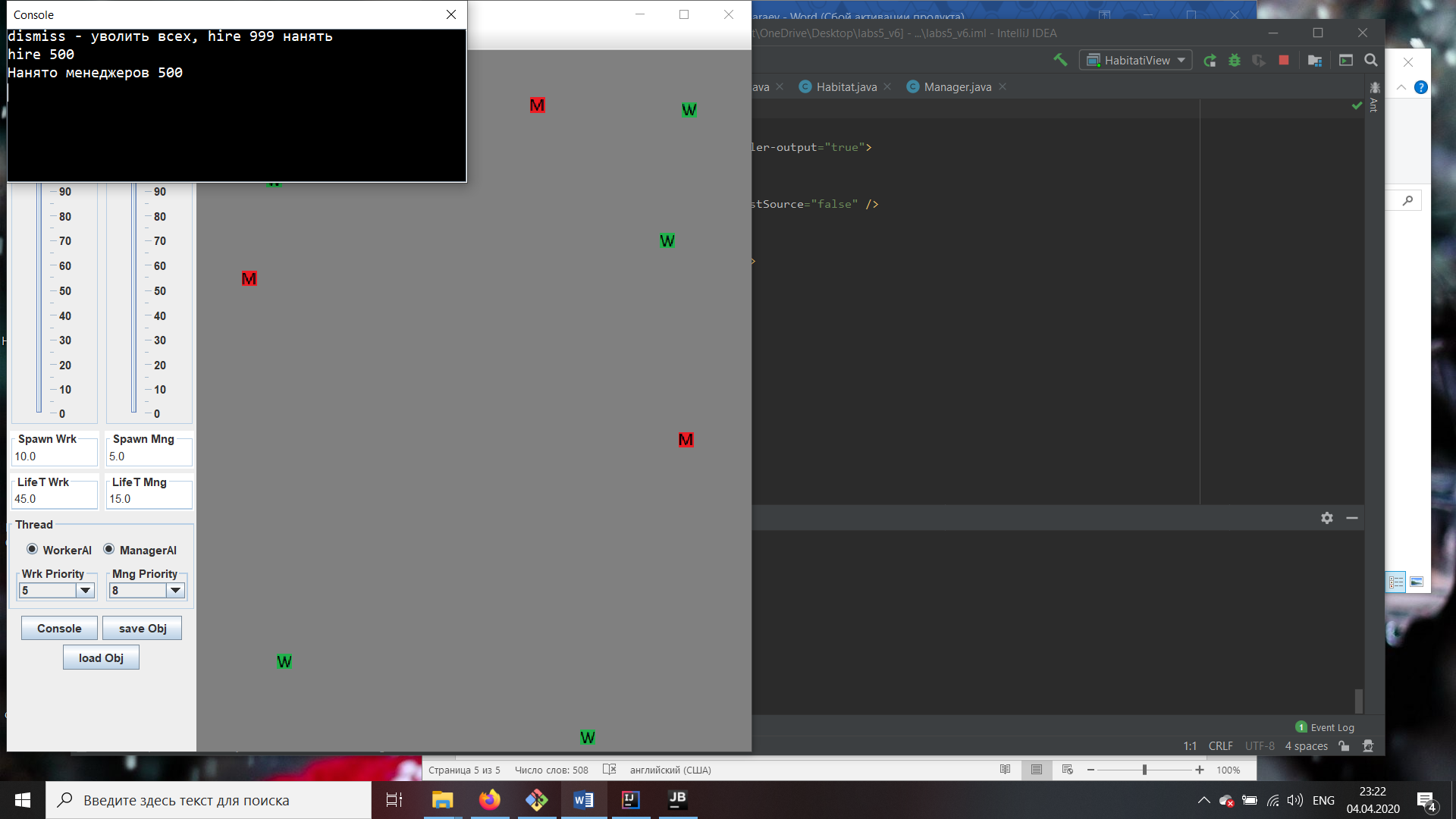
45.0

0.15

5.0

15.0

Окно консоли



**Выводы:** в ходе лабораторной работы познакомилися с принципами работы с потоками ввода-вывода в файл при помощи байтового потока и средствами сериализации java. Изучили основы регулярных выражений для обработки ввода пользователя