Лабораторная работа №1. ПО для работы с Java

Требование к отчету.

Отчет должен содержать:

- Титульный лист согласно образца с указанным номером и названием лабораторной работы.
- Оглавление с указанием номера страницы для каждого раздела.
- Цель работы.
- Теоретическое обоснование. Пишется самостоятельно и должно охватывать вопросы, затрагиваемые в лабораторной работе.
- Задание.
- Документированные листинги программ. После каждого листинга программы должен приводится скриншот с результатом её работы.
- Выводы.

Теоретическое обоснование.

Программное обеспечение

Все необходимое для работы с Java программное обеспечение имеется в открытом (то есть бесплатном) доступе и может быть загружено через Интернет. Оптимальный джентльменский набор размещается на двух сайтах.
Сайт поддержки языка программирования Java www.java.com и сайт www.netbeans.org с бесплатной программой *NetBeans* – интегрированной средой разработки.
Чтобы начать работу с Java, необходимо установить на компьютер систему JDK (аббревиатура от *Java Development Kit*) и интегрированную среду разработки NetBeans.
Описание NetBeans:
https://www3.ntu.edu.sg/home/ehchua/programming/howto/NetBeans_HowTo.html

В состав пакета JDK входит стандартный компилятор (файл javac.exe), стандартные библиотеки классов, примеры программы, документация и исполняющая система Java (комплект JRE - аббревиатура от *Java Runtime Environment*).
Полезная информация:
https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_HowTo.html#Set-JAVA_HOME

Создание консольного приложения

Общая схема реализации процесса введения данных с консоли посредством класса Scanner такова: на основе стандартного потока ввода System.in создается объект класса Scanner, через который и осуществляется консольный ввод. При этом полезными могут оказаться методы класса Scanner, среди которых имеет смысл выделить следующие:

- nextLine() — считывание текстовой строки;
- next() — считывание одного слова;
- nextInt() — считывание целого числа;
- nextDouble() — считывание действительного числа.

Пример использования класса Scanner и его методов для реализации в программе консольного ввода приведен

Пишем консольное приложение:

```
import java.util.*;
public class HelloWorldConsol {
```

```
    public static void main(String[] args) {
        // Объект класса Scanner создается на основе объекта System.in:
        Scanner inp=new Scanner(System.in, “Cp866”);
        String name; // Текстовое поле (имя):
        int age; // Числовое поле (возраст):

        System.out.println("Как Вас зовут?"); // Задаем вопрос:

        name=inp.nextLine();// Считываем текст (имя):

        System.out.println("Добрый день, "+name+"!"); // Приветствие:

        System.out.println("Сколько Вам лет?"); // Задаем вопрос:

        age=inp.nextInt();// Считываем число (возраст):

        System.out.println(name+", Вам "+age+" лет!"); // Вывод сообщения:
}}
```

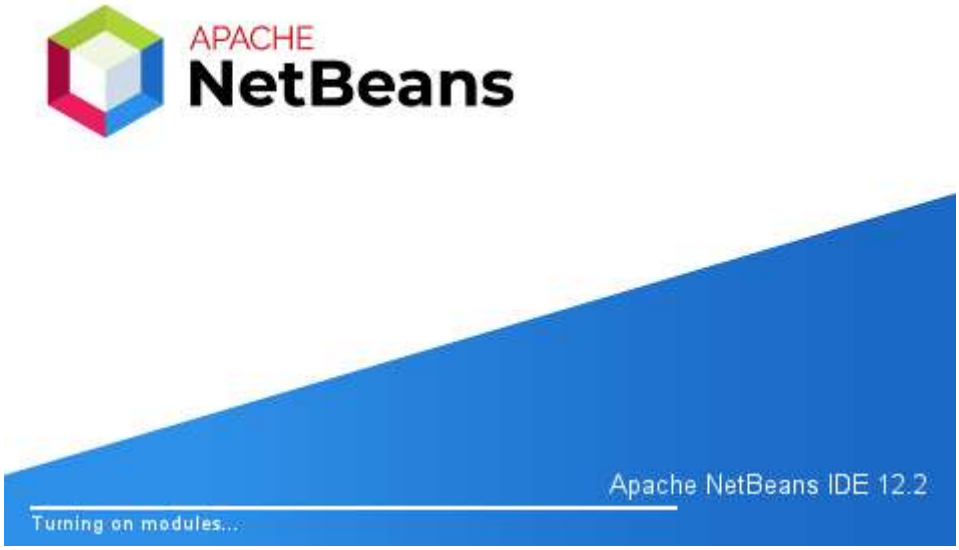
Компилируем и запускаем приложение.

```
javac -encoding utf8 HelloWorldConsol.java
java HelloWorldConsol
```

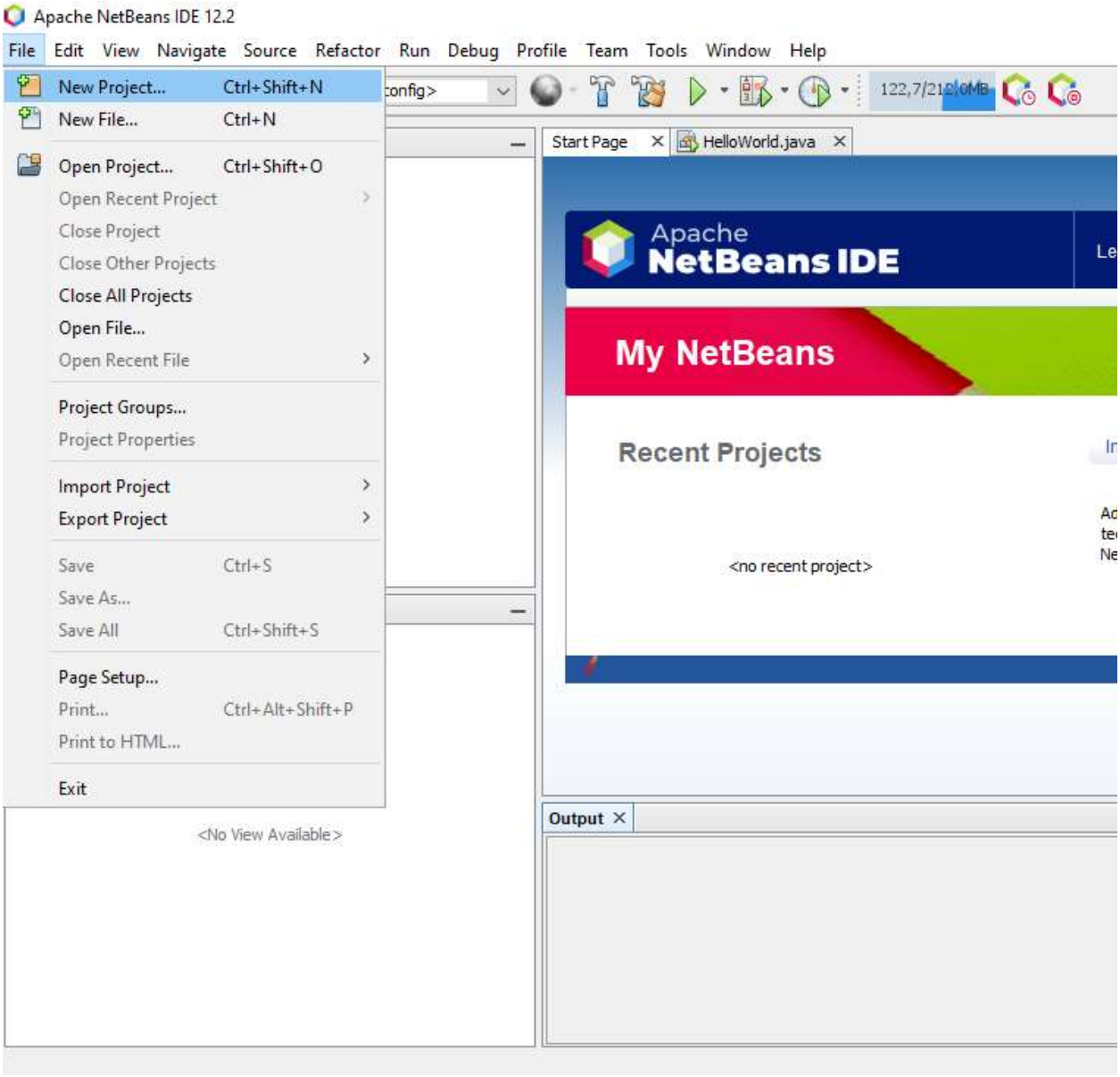
```
C:\1>javac -encoding utf8 HelloWorldConsol.java

C:\1>java HelloWorldConsol
Как Вас зовут?
Ксюша
Добрый день, Ксюша!
Сколько Вам лет?
98
Ксюша, Вам 98 лет!
```

Создание приложения с использованием GUI

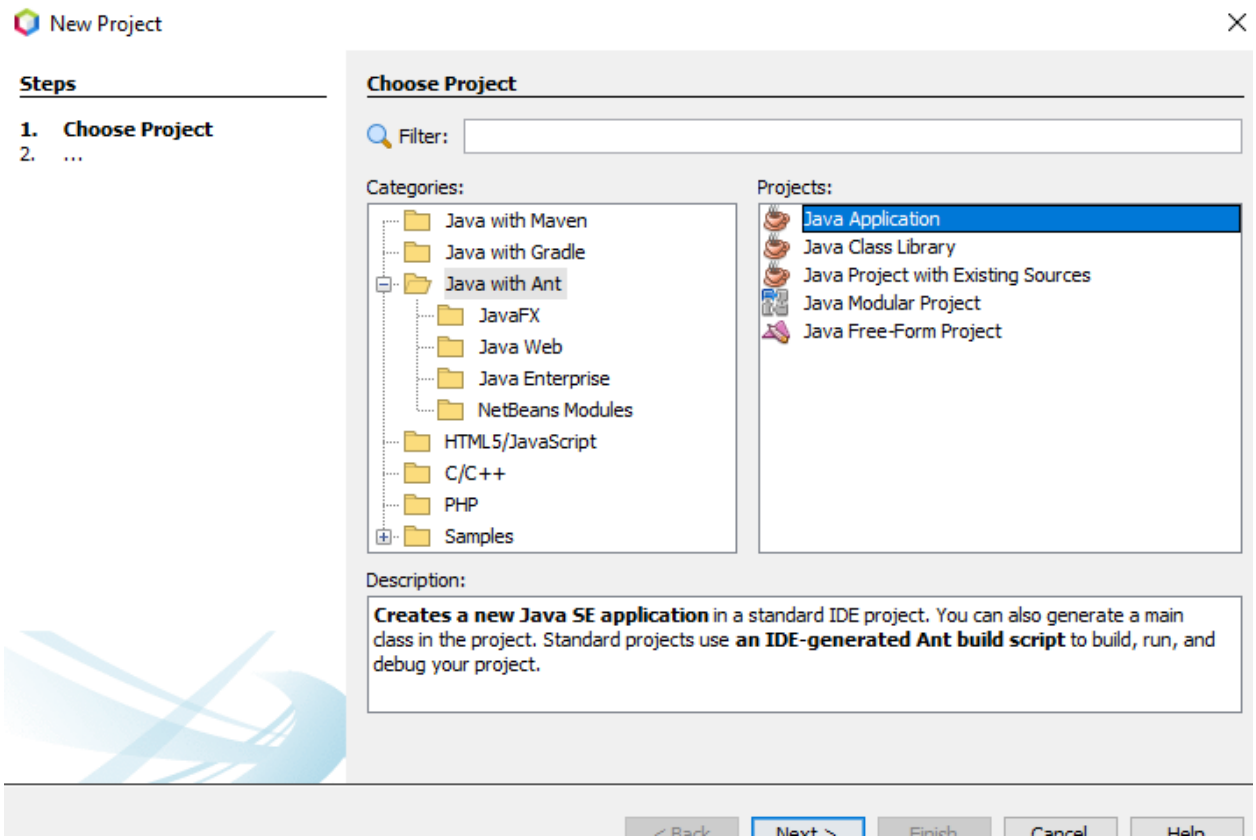
Запускаем приложение NetBeans.



В окне приложения выбираем команду **File_New Project**,

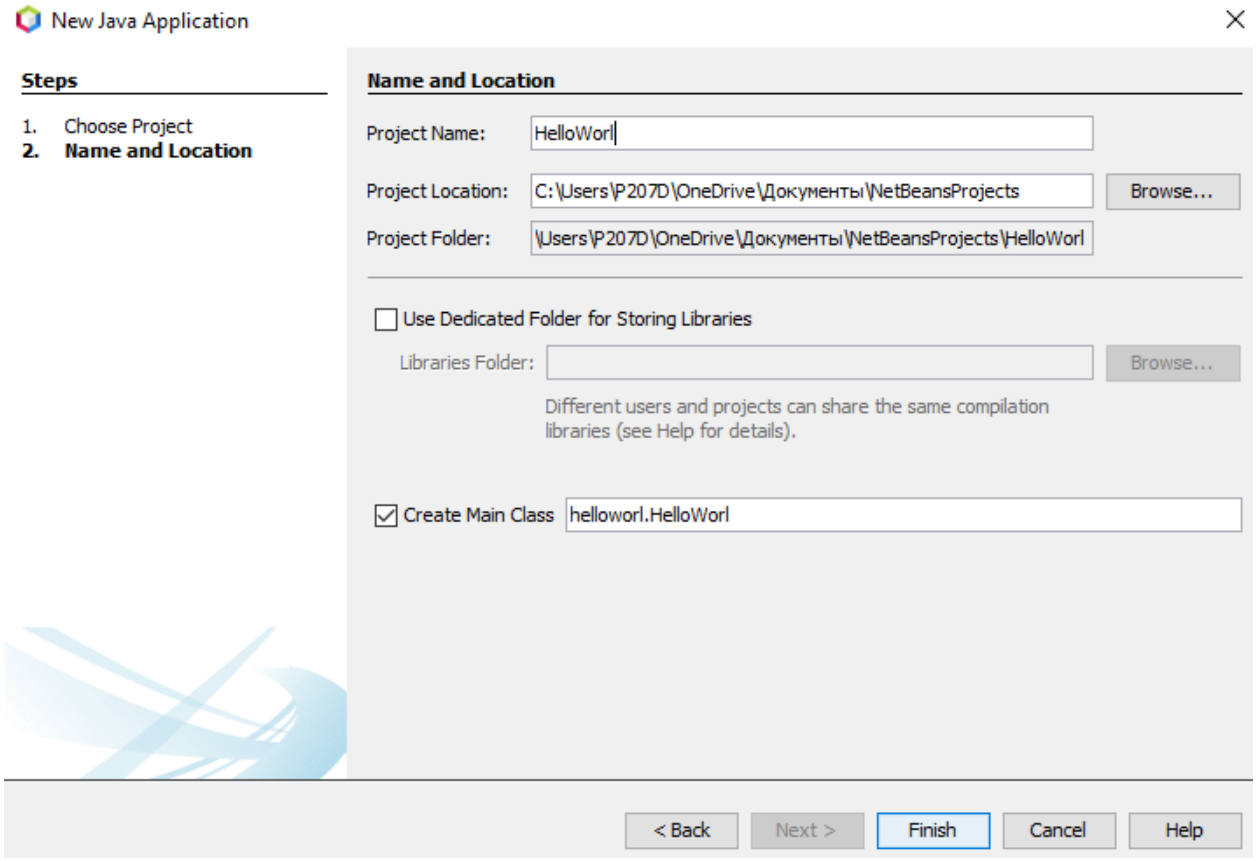


Откроется окно с названием **New Project**, в котором следует выбрать тип создаваемого приложения



Выбираем в категории языков (список **Categories** в левой части окна) пункт **Java** (что соответствует языку Java), а в правой части окна в списке **Projects** выбираем пункт **Java Application**, и щелкаем кнопку **Next**

Появится следующее окно с названием **New Java Application**

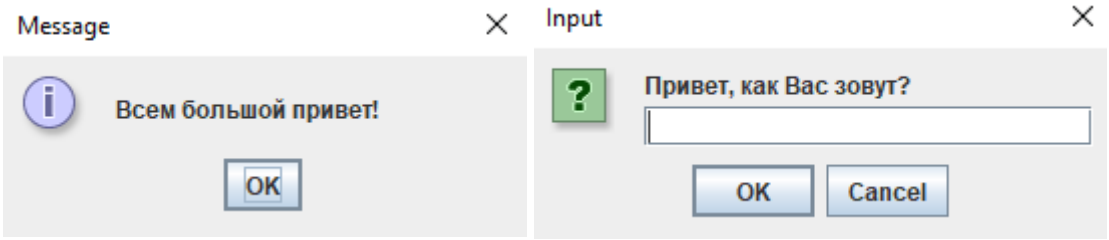


Вводим название и жмем **Finish**.

Пишем графическое приложение:

```
import javax.swing.*;
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        JOptionPane.showMessageDialog(null, "Всем большой привет!");
        String name;
        name=JOptionPane.showInputDialog("Привет, как Вас зовут?");
        JOptionPane.showMessageDialog(null, "Приятно познакомиться, \n"+name+"!");
        JOptionPane.showMessageDialog(null, name);
        byte x=10, y=50;
        x=(byte)(x+1);
        byte z=(byte)(x+y);
        JOptionPane.showMessageDialog(null, z);
    }

}
```

Запускаем и любуемся ходом её выполнения.



.....

Пояснения к коду:

Ключевое слово import является инструкцией подключения библиотеки, в терминологии Java - классов и пакетов. В Java библиотеки реализуются через *классы*, объединенные в *пакеты*. Какой именно ресурс подключается, позволяет определить вторая часть команды. В данном случае диалоговое окно, которое отображается при выполнении программы, описано в библиотеке графических компонентов, которая называется Swing. Это название присутствует в инструкции javax.swing.*. В такого рода инструкциях указывается полный путь к подключаемым средствам. Путь отражает структуру пакетов Java (пакет может содержать классы, *интерфейсы* и другие пакеты). В качестве разделителя при указании структуры пакетов используется точка, а звездочка означает все классы в пакете. Поэтому инструкция import javax.swing.* означает не что иное, как подключение библиотеки Swing.
Вся остальная часть программного кода представляет собой описание класса HelloWorld – основного и единственного класса программы.
Ключевое слово public означает, что класс доступен за пределами того пакета, в котором он описан. Здесь, строго говоря, можно было бы обойтись и без этого ключевого слова, но обычно главный класс программы описывается с ключевым словом public. Ключевое слово class свидетельствует о том, что речь идет о классе, а не о чем-то другом. После ключевого слова class указывается название класса – в данном случае класс называется HelloWorld.
Далее описывается программный код класса,
Внутри программного блока класса, обозначенного фигурными скобками, описан метод main(). Инструкция public static void main(String[] args) называется *сигнатурой* метода, и каждое ключевое слово в ней имеет определенное назначение. Слово public означает, что метод

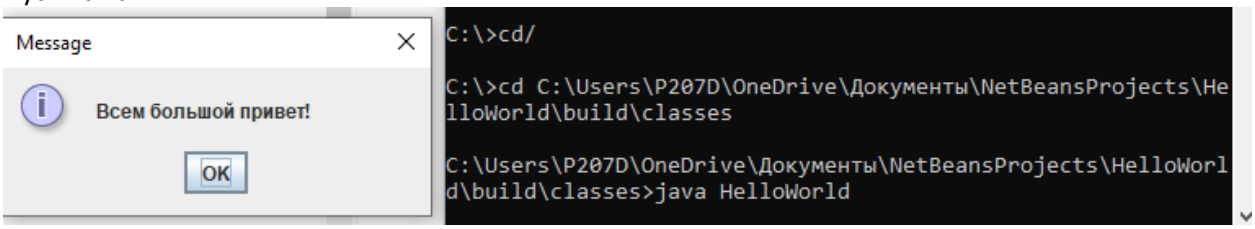
доступен за пределами класса. В данном случае это важно. Выполнение программы означает выполнение метода main(). Очевидно, что метод должен вызываться извне, ведь не может программа запускать сама себя. Именно поэтому метод объявляется как открытый, то есть с ключевым словом public. Следующее ключевое слово static означает, что метод *статический*. То, что метод статический, означает, в свою очередь, что его можно вызывать без создания экземпляра класса, то есть объекта. Стандартная схема вызова обычного, нестатического метода, состоит в том, что на основе класса создается объект, и из объекта вызывается метод. С методом main() такой номер не пройдет в принципе. Ведь для того, чтобы создать объект класса HelloWorld, необходимо сначала запустить программу на выполнение. Это означает, что нужно выполнить метод main(). А если метод нестатический, то для его выполнения необходимо создать экземпляр класса, и так далее – круг замыкается. Следовательно метод main() должен быть статическим.
Ключевое слово void означает, что метод не возвращает результат. То есть метод что-то делает и завершает работу. Все вычисления происходят только внутри метода. После завершения метода от всех вычисленных значений в методе не остается и следа.
У метода main() один аргумент, который называется args (но здесь допускается и другое имя). Этот аргумент текстовый, о чем свидетельствует ключевое слово String. Наличие пустых квадратных скобок [] после ключевого слова String свидетельствует о том, что аргумент метода main() не просто текст, а текстовый массив, то есть набор текстовых значений.
Команда JOptionPane.showMessageDialog(null, "Всем большой привет!"). В этой команде ссылка JOptionPane есть не что иное, как класс библиотеки Swing. Класс JOptionPane отвечает за работу с диалоговыми окнами. В частности, с его помощью можно создавать и отображать диалоговые окна четырех типов:

- Диалоговые окна для вывода информационных сообщений.
- Диалоговые окна получения подтверждения на определенные действия от пользователя.
- Диалоговые окна получения (ввода) данных от пользователя.
- Диалоговые окна для выполнения настроек.

Именно для того, чтобы можно было использовать класс JOptionPane со всеми его утилитами, в начале программы и размещалась команда import javax.swing.*. Если бы мы не использовали класс JOptionPane, без указанной инструкции можно было обойтись.
У класса JOptionPane имеется статический метод showMessageDialog(), которым отображается диалоговое окно информационного типа, то есть окно для вывода информационного сообщения.
Если вызывается статический метод класса, то правило его вызова такое: сначала указывается имя класса, и через точку имя метода с аргументами (если нужно). Метод showMessageDialog() допускает разный способ передачи аргументов. В данном случае методу передается два аргумента. Первым аргументом указано ключевое слово null. Вообще первый аргумент определяет, какое окно порождает отображаемое на экране диалоговое окно с сообщением. Ключевое слово null означает, что такого окна нет – диалоговое окно порождается не окном, а вызывается из метода main(). Второй аргумент метода showMessageDialog() представляет собой текст сообщения в окне. Текст заключается в двойные кавычки.
Команда String name объявляет текстовую *переменную* с названием name.
Следующая команда name=JOptionPane.showInputDialog("Привет, как Вас зовут?");
Окно, которое выводится на экран методом showInputDialog(), имеет поле для ввода текста. Текст, который пользователь вводит в поле, возвращается методом в качестве результата при нажатии клавиши подтверждения, инче null. Этот текст может быть присвоен какой-нибудь переменой. В данном случае значение (текст) записывается в переменную name.
Наконец, командой
JOptionPane.showMessageDialog(null, "Приятно познакомиться, \n"+name+"!");
отображается информационное окно. Такого типа команда уже рассматривалась выше. Особенность этой команды связана со вторым текстовым аргументом метода showMessageDialog(). Он представляет собой *объединение* трех текстовых фрагментов. Для объединения используется оператор сложения +. Если складываются текстовые фрагменты, то происходит их последовательное объединение. В данном случае к тексту "Приятно познакомиться,\n" добавляется значение текстовой переменной name и еще восклицательный

знак (текст "!"). Инструкция \n означает переход к новой строке. Поэтому полученное в результате объединения трех текстовых фрагментов сообщение будет отображаться в диалоговом окне в две строки.
Далее следуют ещё две команды отображения информационного окна.

Запуск скомпилированной Java программы (HelloWorld.class) из командной строки:
Пуск->cmd



Для компиляции и запуска программы через CMD следует использовать команды:

```
C:\1>javac -encoding utf8 HelloWorld.java

C:\1>java HelloWorld
```

Здесь предполагается, что исходный файл **HelloWorld.java** находится в папке 1 на диске C.
Ключ `-encoding utf8` указывает на кодировку исходного файла.

Задание.

1. Установить ПО для работы с Java.
2. Написать тестовые консольное и GUI приложения, выполняющие прием данных с клавиатуры и выводящих их на экран.