**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ**

**им. проф. М.А. Бонч-Бруевича»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций**

**Отчет о выполнении  
практического занятия №11-12-13**

Выполнил: Обучающиеся 4 курса, 581 группы,

Филипович Валерий Анатольевич

Состав бригады: -

Проверил: преподаватель  
Баталов Дмитрий Иннокентьевич

Санкт-Петербург

2021 г.

**Постановка задачи**

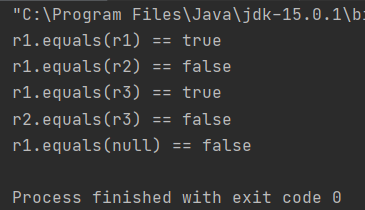
Реализовать два консольных приложения со следующими условиями:

* **Задание 1.** Запустить на выполнение код программ, приведенных в конспекте лекционного занятия «Занятие № 10. Пакет java.lang». Выполнять только те программы, для которых приведены примеры вывода на консоль. Обращать внимание на возникающие ошибки компиляции, которые нужно исправить. Добиться успешного выполнения программ.
* **Задание 2.** Запустить на выполнение код программ, приведенных в конспекте лекционного занятия «Занятие № 11. Пакет java.util». Выполнять только те программы, для которых приведены примеры вывода на консоль. Обращать внимание на возникающие ошибки компиляции, которые нужно исправить. Добиться успешного выполнения программ.

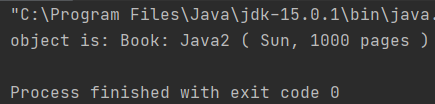
**Ход выполнения работы**

1. Создал новый консольный проект «ObjectEqualsProgram»;
2. Добавил в код фрагмент, в котором метод equals() у класса Rectangle был переопределен таким образом, чтобы прямоугольники были равны, если их можно наложить друг на друга (геометрическое равенство);
3. Создал новый консольный проект «ObjectBookProgram»;
4. Добавил в код фрагмент, в котором продемонстрировано то, что в классах-наследниках метод toString() может быть переопределен для получения более наглядного описания объекта. Обычно это значения некоторых полей, характеризующих экземпляр.
5. Создал новый консольный проект «ClassVehicleProgram»;
6. Добавил в код фрагмент, в котором продемонстрировано использование технологии reflection.
7. Создал новый консольный проект «IntegerProgram»;
8. Добавил в код фрагмент, в котором продемонстрирована работа методов класса-обертки Integer.
9. Создал новый консольный проект «StringProgram»;
10. Добавил в код фрагмент, в котором продемонстрирована работа методов класса String;
11. Создал новый консольный проект «StringConcatProgram»;
12. Добавил в код фрагмент, в котором продемонстрирована работа метода конкатенации concat() класса String;
13. Создал новый консольный проект «StringBufferProgram»;
14. Добавил в код фрагмент, в котором продемонстрирована работа методов класса StringBuffer;
15. Создал новый консольный проект «StringInsertProgram»;
16. Добавил в код фрагмент, в котором продемонстрировано то, что методы insert() и append() возвращают сам объект, у которого вызываются;
17. Создал новый консольный проект «StringBufferChangeProgram»;
18. Добавил в код фрагмент, в котором продемонстрирована то, что класс StringBuffer изменяемый;
19. Создал новый консольный проект «SystemClassProgram»;
20. Добавил в код фрагмент, в котором продемонстрирована работа методов класса System;
21. Создал новый консольный проект «SetProgramFirst»;
22. Добавил в код фрагмент, в котором продемонстрирована работа метода set() класса Calendar;
23. Создал новый консольный проект «SetProgramSecond»;
24. Добавил в код фрагмент, в котором продемонстрирована работа метода set() класса Calendar: считывание даты не производится, соответственно, ее вычисление не выполняется до тех пор, пока все поля не установлены;
25. Cоздал новый консольный проект «AddProgramFirst»;
26. Добавил в код фрагмент, в котором продемонстрирована работа метода add() класса Calendar;
27. Cоздал новый консольный проект «RollProgram»;
28. Добавил в код фрагмент, в котором продемонстрирована работа метода roll() класса Calendar;
29. Cоздал новый консольный проект «TimeZone»;
30. Добавил в код фрагмент, в котором продемонстрирована работа методов класса TimeZone;
31. Cоздал новый консольный проект «SimpleTimeZone»;
32. Добавил в код фрагмент, в котором продемонстрирована работа методов класса SimpleTimeZone;
33. Cоздал новый консольный проект «ObservableProgram»;
34. Добавил в код фрагмент, в котором продемонстрирована работа методов класса Observable и интерфейса Observe;
35. Cоздал новый консольный проект «ArrayListProgram»;
36. Добавил в код фрагмент, в котором продемонстрирована работа методов класса ArrayList;
37. Cоздал новый консольный проект «LinkedListProgram»;
38. Добавил в код фрагмент, в котором продемонстрирована работа методов класса LinkedList;
39. Cоздал новый консольный проект «HashMapProgram»;
40. Добавил в код фрагмент, в котором продемонстрирована работа методов класса HashMap;
41. Cоздал новый консольный проект «TreeMapProgram»;
42. Добавил в код фрагмент, в котором продемонстрирована работа методов класса TreeMap;
43. Cоздал новый консольный проект «CollectionsProgram»;
44. Добавил в код фрагмент, в котором продемонстрирована работа методов класса Collections;
45. Cоздал новый консольный проект «PropertiesProgram»;
46. Добавил в код фрагмент, в котором продемонстрирована работа методов класса Properties;
47. Cоздал новый консольный проект «StringTokenizerProgram»;
48. Добавил в код фрагмент, в котором продемонстрирована работа методов класса StringTokenizer;
49. Cоздал новый консольный проект «BitSetProgram»;
50. Добавил в код фрагмент, в котором продемонстрирована работа методов класса BitSet;
51. Cоздал новый консольный проект «RandomProgram»;
52. Добавил в код фрагмент, в котором продемонстрирована работа методов класса Random;
53. Cоздал новый консольный проект «LocaleProgram»;
54. Добавил в код фрагмент, в котором продемонстрирована работа методов класса Locale;

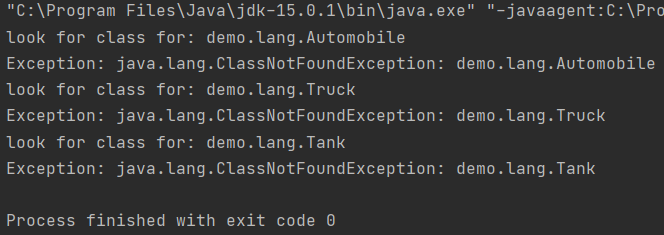
**Результат выполнения программы**



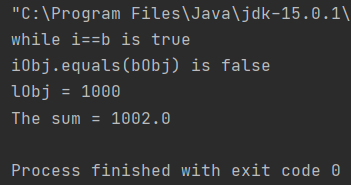
*Рис. 1. Результат работы программы «ObjectEqualsProgram»*



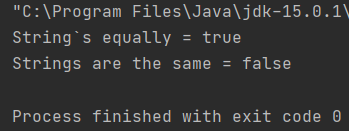
*Рис. 2. Результат работы программы «ObjectBookProgram»*



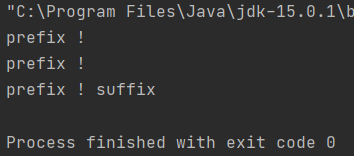
*Рис. 3. Результат работы программы «ClassVehicleProgram»*



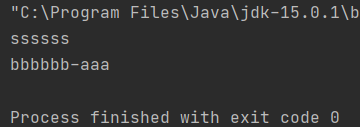
*Рис. 4. Результат работы программы «IntegerProgram»*



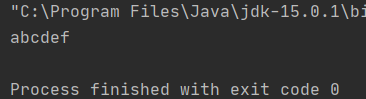
*Рис. 5. Результат работы программы «StringProgram»*



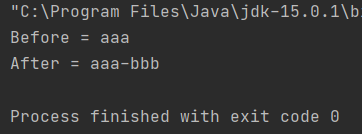
*Рис. 6. Результат работы программы «StringConcatProgram»*



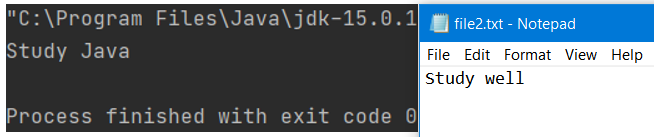
*Рис. 7. Результат работы программы «StringBufferProgram»*



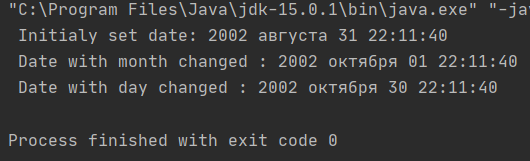
*Рис. 8. Результат работы программы «StringInsertProgram»*



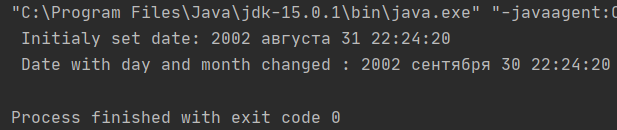
*Рис. 9. Результат работы программы «StringBufferChangeProgram»*



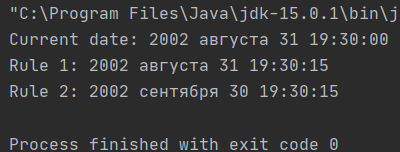
*Рис. 10. Результат работы программы «SystemClasProgram»*



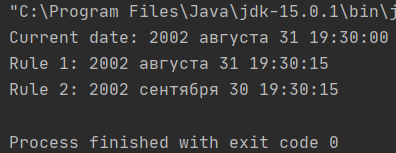
*Рис. 11. Результат работы программы «SetProgramFirst»*



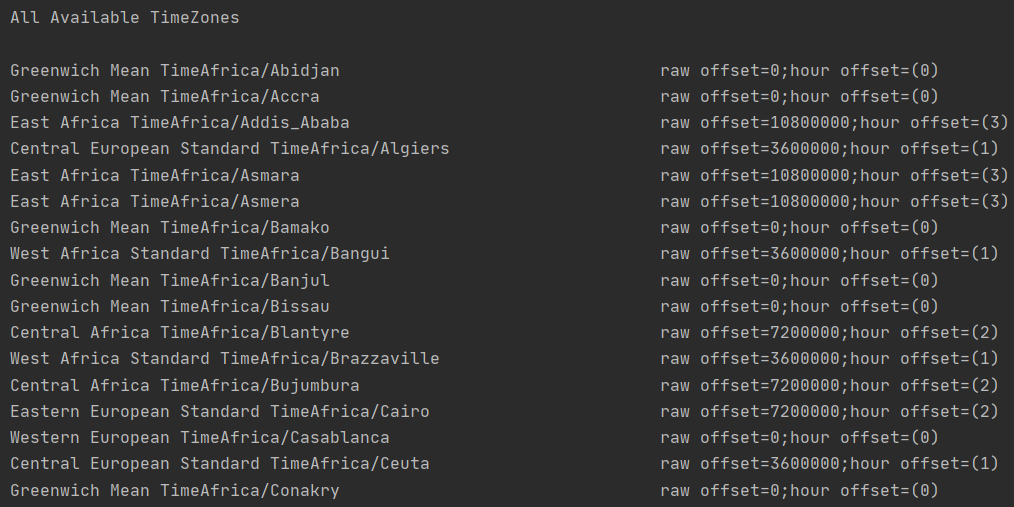
*Рис. 12. Результат работы программы «SetProgramSecond»*



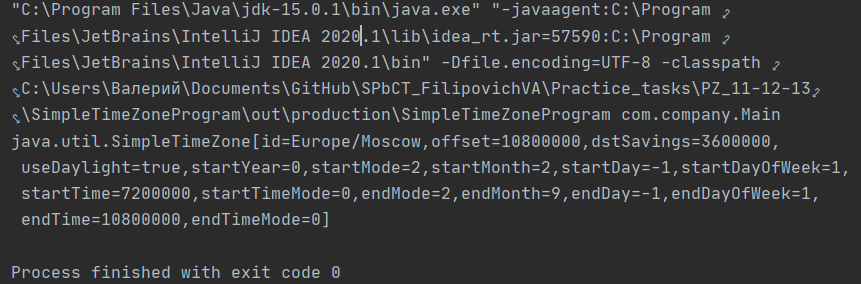
*Рис. 13. Результат работы программы «AddProgramFirst»*



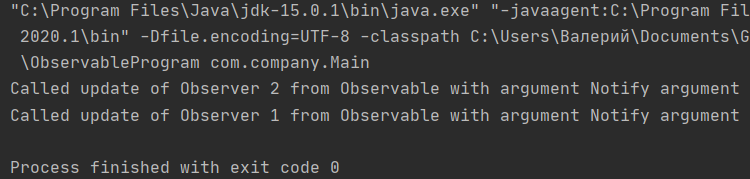
*Рис. 14. Результат работы программы «RollProgram»*



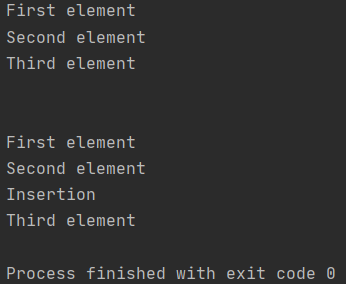
*Рис. 15. Результат работы программы «TimeZoneProgram»*



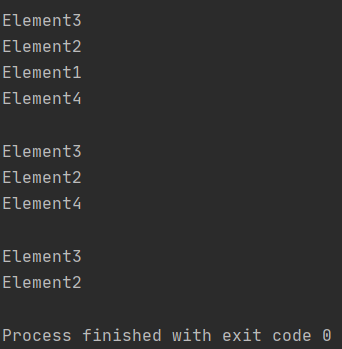
*Рис. 16. Результат работы программы «SimpleTimeZoneProgram»*



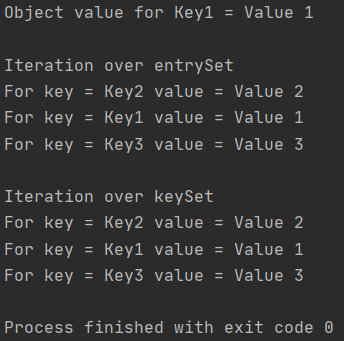
*Рис. 17. Результат работы программы «ObservableProgram»*



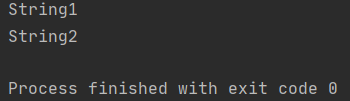
*Рис. 18. Результат работы программы «ArrayListProgram»*



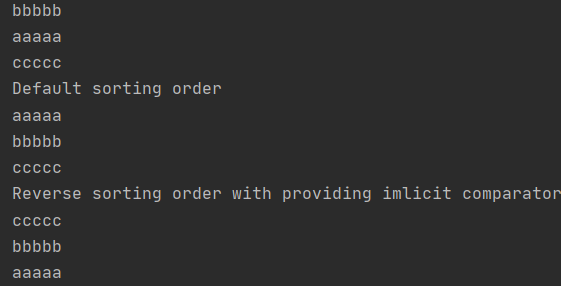
*Рис. 19. Результат работы программы «LinkedListProgram»*



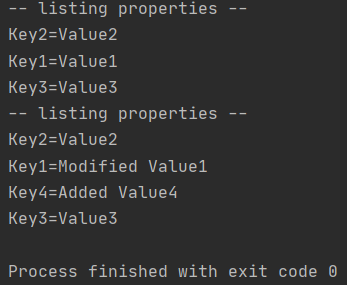
*Рис. 20. Результат работы программы «HashMapProgram»*



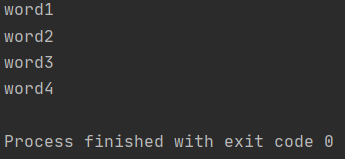
*Рис. 21. Результат работы программы «TreeMapProgram»*



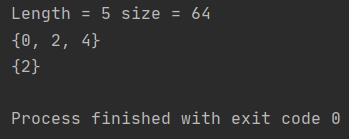
*Рис. 22. Результат работы программы «CollectionsProgram»*



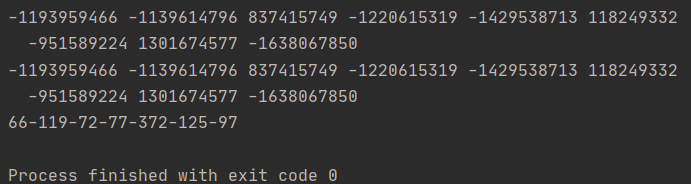
*Рис. 23. Результат работы программы «PropertiesProgram»*



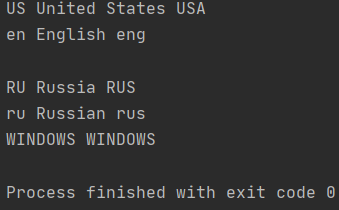
*Рис. 24. Результат работы программы «StringTokenizerProgram»*



*Рис. 25. Результат работы программы «BitSetProgram»*



*Рис. 26. Результат работы программы «RandomProgram»*



*Рис. 27. Результат работы программы «LocaleProgram»*

**Ссылка на GitHub**

<https://github.com/ValeriyFilipovich/SPbCT_FilipovichVA/tree/master/Practice_tasks/>

**Ответы на контрольные вопросы**

1. ***Какие основные составные части должны присутствовать в каждой Java-программе, их функции (назначение)?***

Все Java-программы содержат в себе четыре разновидности блоков: классы (classes), методы (methods), переменные (variables) и пакеты (package).

Методы есть не что иное как функции или подпрограммы. В переменных же хранятся данные. Данные понятия присутствуют так или иначе во всех языках программирования. С другой стороны, классы представляют собой фундамент объектно-ориентированных свойств языка. Для простоты на данном этапе изучения языка Jаva можно сказать, что класс — это некое целое, содержащее переменные и методы. Пакеты содержат в себе классы и помогают компилятору найти те классы, которые нужны ему для компиляции пользовательской программы. Классы, входящие в один пакет, особым образом зависят друг от друга, пока же, опять-таки для простоты, можно рассматривать просто как наборы классов. Например, приложение Hello импортирует пакет java.util, в котором содержится класс Date.

1. ***Что такое первичный класс приложения? Какой обязательный метод он должен содержать?***

Java-программа может содержать в себе любое количество классов, но один из них всегда имеет особый статус и непосредственно взаимодействует с оболочкой времени выполнения (первичный класс). В таком классе обязательно должны быть определены один (для приложений) или несколько (для апплетов) специальных методов. Для приложений первичный класс должен обязательно содержать метод main().

1. ***Какие существуют виды переменных Java, чем они отличаются друг от друга?***

В Java существует два вида переменных. Первый - примитивные типы (primitive types). Ко второму виду переменных - ссылочные типы (reference type).

Примитивные и ссылочные типы различаются по тому, как переменные этих типов передаются в качестве параметров методам (то есть функциям). Переменные примитивных типов передаются по значению, тогда как ссылочные переменные всегда передаются по ссылке.

1. ***Какие примитивные типы определены в Java, особенности булевского типа?***

Всего в Java определено восемь примитивных типов: int (4b), short (2b), byte (1b), long (8b), float (4b), double (8b), boolean (true, false), char (2b).

Булевские (логические, двоичные) переменные могут иметь одно из двух допустимых значений: true или false. Двоичные константы являются именно константами, а не строками. Их нельзя преобразовать в строковый тип. Они также не являются целыми значениями нуль или единица. Булевским переменным можно присваивать не только булевские константы, но и результаты сравнения переменных различных типов. Операции !, !=, == работают с булевскими значениями.

1. ***Что называется процессом реализации ссылочного типа?***

Когда для переменной ссылочного типа выделяется память при помощи оператора new, то тем самым этот ссылочный тип реализуется. Таким образом, каждая переменная ссылочного типа является реализацией, объектом или экземпляром соответствующего типа.

1. ***Что делает конструктор класса? Должен ли он обязательно явно присутствовать в объявлении класса?***

Чтобы определить, какие еще действия должны выполняться в момент реализации класса, нужно определить так называемый конструктор (constructor) - метод, имеющий такое же имя как и класс. В классе может быть не один конструктор, они должны отличаться списком аргументов. Конструкторы можно использовать для инициализирующих действий, например для присвоения начальных значений. Можно использовать все определенные в классе конструкторы (с параметрами и без них), вызывая их при создании объектов класса. Если в классе отсутствует определение конструктора без параметров, то вызывается конструктор без параметров по умолчанию.

1. ***Какие существуют виды ссылочных типов?***

Ссылочные типы отличаются от примитивных тем, что они не определены в самом языке Java, и поэтому количество памяти, требуемое переменных этих типов, заранее знать невозможно. Пример одного из ссылочных типов - это тип массива. Массивы языка Java могут состоять из переменных любых типов, включая типы, определенные пользователем (классы).

1. ***Что такое типы, определенные пользователем?***

Это такие ссылочные типы, как классы или интерфейсы, которые определил (или выбрал из существующих) сам пользователь.

1. ***Что такое стандартные типы, определенные пользователем?***

Название “типы, определенные пользователем” не подразумевает, что каждый пользователь сам должен определять для себя все типы. В состав интерфейса прикладного программирования (Application Programming Interface -API) входят десятки стандартных классов, которые можно использовать в своих программах.

1. ***В чем особенности строковых переменных?***

Данный класс представляет собой гибрид примитивных и ссылочных типов. В основе своей тип String является типом, определенным пользователем, так как он определяется как одноименный класс String, содержащий в себе методы и переменные. Но в то же время этот тип проявляет некоторые свойства примитивного типа, что выражается, в частности, в том, как осуществляется присвоение значение этим переменным при помощи знака операции =.

1. ***Чем массивы Java отличаются от массивов других языков, их преимущества?***

Оператор new запрашивает для массива необходимую память. При создании ссылочной переменной типа массив не нужно объявлять размер массива, это делается при реализации типа массива при помощи оператора new. Доступ же к элементам массива ничем не отличается от доступа к элементам массива в C/C++.

Массивы Java имеют три важных преимущества перед массивами других языков. Во-первых, программисту не обязательно указывать размер массива при его объявлении. Во-вторых, любой массив в языке Java является переменной - а это значит, что его можно передать как параметр методу и использовать в качестве значения, возвращаемого методом. И в-третьих, не составляет никакого труда узнать, каков размер данного массива в любой момент времени через специальную переменную length, имеющуюся в каждом массиве.

1. ***Как переменные различных видов передаются в качестве параметров методам?***

В качестве параметров в языке Java можно передавать переменные любого типа, включая типы, определенные через классы, и массивы переменных любого типа и размера. Однако в качестве параметров переменные примитивных типов ведут себя иначе, чем переменные ссылочных типов.

Все переменные примитивных типов передаются методам по значению (by value). Это означает, что в момент вызова метода делается копия переменной, передаваемой методу. Если метод в своем теле будет изменять значение переданной ему в качестве параметра переменной, то содержимое исходной переменной изменяться не будет, так как все действия будут производится с ее копией.

Напротив, значения переменных ссылочного типа, переданных в качестве параметров, можно изменить в теле метода. Когда методу в качестве параметра передается переменная ссылочного типа, то при изменении ее значения явным образом меняется содержимое того, на что указывает эта переменная.

1. ***Как ведут себя строковые переменные при передаче их в качестве параметров?***

Несмотря на то, что тип строковых переменных (тип String) является определенным пользователем типом, он не ведет себя как ссылочный тип при передаче параметров. Переменные типа String в качестве параметров всегда передаются по значению, - то есть метод, получив строковую переменную в качестве параметра, в своем теле будет фактически работать с копией этой строковой переменной. Иначе говоря, изменение строковой переменной в теле метода не влияет на значение этой же переменной снаружи метода.

1. ***Что такое совмещение методов?***

Иногда возникает необходимость создавать две или несколько функций, выполняющих, по сути, одни и те же действия, но имеющие различные списки параметров. В языке Java можно присвоит одно и то же имя нескольким методам, которые различаются списками своих параметров. Этот процесс называется совмещением или перегрузкой методов.

1. ***Что такое элементы класса и элементы экземпляра класса, чем они отличаются друг от друга? Как нужно указывать, что переменная или метод является элементом класса, а не экземпляра?***

В Java переменные и методы класса могут быть либо элементами класса, либо элементами экземпляра (объекта) класса, в котором они объявлены, что определяется присутствием или отсутствием модификатора static.

Элементы класса, будучи уникальными в своем классе, используются всеми объектами, созданными из этого класса, - то есть все объекты, созданные из данного класса, разделяют статические переменные и методы, определенные в этом классе (для всех объектов существует только один экземпляр статической переменной). Элементы экземпляра класса, с другой стороны, создаются каждый раз, когда создается объект.

Чтобы обратиться к методам и переменным экземпляра (динамическим элементам), объект надо сначала реализовать из класса, после чего можно получить к ним доступ, используя синтаксис (запись с точкой). А для использования статических методов и переменных (элементов класса) объект можно не создавать.

1. ***Для чего используются модификаторы доступа? Какие существуют модификаторы доступа, как они ограничивают доступ к элементам?***

Модификаторы доступа используются для управления доступностью элементов класса из других частей программы (в других классах). Применение ключевых слов public (открытый), protected (защищенный), private (закрытый) к элементам помогает управлять способностью других объектов пользоваться ими.

1. ***Что позволяет делать процесс наследования?***

Наследование (inheritance) упрощает практическое использование классов, так как позволяет расширять уже написанные и отлаженные классы, добавляя к ним новые свойства и возможности.

1. ***Что такое суперкласс и подкласс?***

Например, вместо создания класса “с нуля”, можно значительно выиграть во времени, используя механизм наследования переменных и методов, определенных в другом классе, обладающем основными необходимыми свойствами. Таким образом создается то, что называется подклассом первоначального класса. Класс, который при этом наследуется (расширяется), называется суперклассом.

1. ***Что такое повторное использование кода?***

При расширении какого-либо класса имеется возможность использования всех написанных и отлаженных методов этого класса - их не нужно создавать заново для подкласса. Это свойство, называемое повторным использованием кода (code reuse), является одним из главных преимуществ объектно-ориентированного програмирования.

1. ***Какие заранее определенные переменные содержит каждый класс Java?***

Каждый класс Java содержит три заранее определенные переменные, которыми можно пользоваться: null, this, super. Первые две переменные относятся к типу Object. Коротко говоря, null представляет собой несуществующий объект, this указывает на текущий экземпляр, переменная super разрешает доступ к открытым переменным и методам, определенным в суперклассе.

1. ***Что можно сделать при помощи переменной this? Что можно сделать при помощи переменной super?***

Иногда бывает необходимо передать другому методу ссылку на текущий объект. Это можно сделать, просто передав переменную особую ссылочную переменную this.

Другая специальная переменная языка Java super ссылается на суперкласс объекта. При помощи нее можно получить доступ к затененным переменным и замещенным методам родительского класса.

1. ***Что такое скрытие переменной, затемнение переменной и замещение метода?***

Затенение переменной класса в его подклассе возникает при объявлении в подклассе переменной таким же именем, что и имя переменной его суперкласса.

Если на уровне класса объявлена переменная-элемент с именем ИмяПеременной, то ничто не мешает объявить локальную переменную с тем же именем ИмяПеременной внутри какого-либо из методов класса, которая скроет вышестоящую переменную и сделает ее недоступной (скрытие переменной).

При замещении метода в классе создается метод, имеющий то же самое имя и список параметров, что и метод в суперклассе. Таким образом метод суперкласса замещается методом подкласса.

1. ***Как импортировать классы из пакетов?***

Существует ряд способов доступа к классам в пакетах, основным из которых является импортирование пакета в начале программ:

import ИмяПакета .\* или import ИмяПакета.ИмяКлассаПакета

1. ***Как использовать явные ссылки на классы из пакетов?***

Если пакет импортирован, то его классы могут использоваться столько раз, сколько требует программа. В случае, когда класс необходимо использовать только один раз, можно явно сослаться на класс, не импортируя пакет, частью которого он является. Для этого перед именем используемого класса просто указывается имя пакета, в котором находится этот класс.

1. ***Как добавить класс в пакет?***

Для создания пакета (т.е. добавления класса в пакет), просто используется следующее предложение в качестве первой строки программы (даже раньше предложения import) в исходном тексте программы:

package ИмяПакета

1. ***Как загрузить графическое изображение из файла?***

Для загрузки изображения из файла используется метод getImage(), принимающий URL это файла. Этот метод может импортировать изображения любого графического формата, поддерживаемого WWW-броузером (чаще всего используются форматы GIF и JPEG).

1. ***Как ускорить вывод графических изображений, загружаемых из файла?***

Для ускорения вывода изображений в процессе вывода большинство апплетов использует двойную буферизацию изображения - сначала загружая изображение в оперативную память, а затем выводя его в окне апплета за один шаг.

1. ***Что называется событием, когда они генерируются?***

Событие — это информация, сгенерированная в ответ на некоторые действия пользователя (перемещение мыши, нажатие клавиши мыши или клавиши на клавиатуре). События также могут быть сгенерированы в ответ на изменение среды - к примеру, когда окно апплета заслоняется другим окном.

1. ***Какие обработчики событий используются чаще всего?***

Для обработки событий от кнопок мыши используется тип MouseListener. На самом деле это интерфейс. Для обработки этих событий надо, чтобы класс реализовывал указанный интерфейс, то есть в классе должны быть реализованы методы этого интерфейса. Если в интерфейсе более одного метода, то можно использовать класс адаптера, который реализует указанный интерфейс, от этого класса-адаптера можно наследовать какой-то класс, а в нем переопределить только тот метод, который нужен.

1. ***Как переменные содержит класс Event для идентификации события?***

Класс ActionEventопределяет четыре целочисленные константы, которые можно использовать для идентификации события: alt\_mask, ctrl\_mask, me-ta\_maskи shift\_mask. Кроме того, существует целочисленная константа, action\_performed, которую можно применять для идентификации action-события.

1. ***Как обработать любое событие?***

Если вы хотите обработать любое событие типа AWTEvent, то вам надо переопределить метод processEvent (), а если более конкретное событие, например, событие клавиатуры, — переопределить более конкретный метод processKeyEvent о. Если вы не переопределяете весь метод целиком, то не забудьте в конце обратиться к методу суперкласса, например, super.processKeyEvent(e);

1. ***Как устранить мерцание при выводе изображений?***

Для устранения эффекта мерцания изображений в процессе вывода большинство апплетов использует двойную буферизацию изображения - сначала загружая изображение в оперативную память, а затем выводя его в окне апплета за один шаг.