Список вопросов к экзамену по предмету Программирование на языкеДжава зима 2020-2021 год

Тема 1. Особенности платформы Java. Синтаксис языка Java

1. **К какому типу языков относится язык Джава**

Java — строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (далее приобретённой Oracle). Каждая переменная и константа представляет определенный тип, который строго определен. Тип данных определен диапазоном значений, которые может хранить переменная или константа. Все выражения и параметры javac проверяет на соответствие типов.
Java - интерпретируемый язык, т.е. написанный код компилируется в байт-код Java, который передается в интерпретатор (JVM)
Т.е. следует принципу "Писать один раз, запускать где угодно" - скомпилированный код Java может работать на всех платформах, поддерживающих Java, без необходимости перекомпиляции.

2. **Особенности языка Джава**

простота, объектная ориентированность и понятность;
надёжность и безопасность;
переносимость и независимость от платформы;
высокая производительность;
интрепретируемость, поточность и динамичность.

3. **Класс Scanner и его использование для чтения стандартного потока вводы**

Этот класс пригодится, если тебе нужно будет считывать данные, которые вводят юзеры. Класс - java.util.Scanner. Его функциональность очень проста. Он, словно настоящий сканер, считывает данные из источника, который ты для него укажешь. Например, из строки, из файла, из консоли. Далее он распознает эту информацию и обрабатывает нужным образом.

4. ***Класс Scanner, конструктор класса Scanner для чтения стандартного потока вводы***

5. **Методы класса Scanner nextLine(), nextInt(), hasNextInt(), hasNextLine() и их использование для чтения ввода пользователя с клавиатуры**

Метод nextLine() обращается к источнику данных, находит там следующую строку, которую он еще не считывал и возвращает ее. После чего мы выводим ее на консоль.
Метод **nextInt()** считывает и возвращает введенное число.
hasNextInt() — метод проверяет, является ли следующая порция введенных данных числом, или нет (возвращает, соответственно, true или false).
hasNextLine() — проверяет, является ли следующая порция данных строкой.
hasNextByte(), **hasNextShort()**, **hasNextLong()**, **hasNextFloat()**, **hasNextDouble()** — все эти методы делают то же для остальных типов данных.

6. **Примитивные типы данных, объявление и присваивание переменных**

Byte, short, int, long, float, double, char, boolean

С помощью перегрузки метода out.println можно выводить в консоль все примитивные типы данных.

7. Условные операторы, полное и неполное ветвление в Джава, синтаксис

Условный оператор if позволяет задать условие, в соответствии с которым дальнейшая часть программы может быть выполнена.
Условный оператор if-else в языке означает «в ином случае». То есть если условие if не является истинным, выводим то, что в блоке else.
Вложенный if служит для проверки сразу на несколько условий, где используются разные действия.
Пример:

```
---------------------------------------------------------------
int a = 20;
int b = 5;

if(a == 20){
   System.out.println("a = 20");
   if(b == 5){
      System.out.println("b = 5");
   }
}
---------------------------------------------------------------
```

«Элвис» или оператор тернар - ?: . Получил прозвище за схожесть с причёской короля рок-н-ролла, он требует три операнда и позволяет писать меньше кода дляя простых условий.
Пример:

```
---------------------------------------------------------------
int a = 20;
int b = 5;

String answer = (a > b) ? "Условие верно" : "Условие ошибочно";
System.out.println(answer);
---------------------------------------------------------------
```

Условный оператор или оператор множества switch позволяет сравнить переменную как с одним, так и с несколькими значениями.
Пример:

```
---------------------------------------------------------------
switch(выражение) {
   case значение1:
      // Блок кода 1
      break;
   case значение2:
      // Блок кода 2
      break;
   case значениеN:
      // Блок кода N
      break;
   default :
```

```
    // Блок кода для default
}
```

---

8. Оператор множественного выбора в Джава, синтаксис

Условный оператор или оператор множества switch позволяет сравнить переменную как с одним, так и с несколькими значениями.

Пример:

```
-----------------------------------------------------------------
switch(выражение) {
   case значение1:
      // Блок кода 1
      break;
   case значение2:
      // Блок кода 2
      break;
   case значениеN:
      // Блок кода N
      break;
   default :
      // Блок кода для default
}
-----------------------------------------------------------------
```

9. Класс System. Работа со стандартами потоками вывода

Системный является одним из базовых классов в Java и принадлежит пакету java.lang. Класс System является финальным и не предоставляет общедоступных конструкторов. Из-за этого все члены и методы, содержащиеся в этом классе, являются статическими по природе.

Таким образом, вы не можете наследовать этот класс для переопределения его методов. Поскольку класс System .имеет множество ограничений, существуют различные предварительно созданные поля и методы класса. Ниже я перечислил несколько важных функций, поддерживаемых этим классом:

Стандартный ввод и вывод.

Ошибка вывода потоков.

Доступ к внешним свойствам и переменным среды.

Встроенная утилита для копирования части массива.

Предоставляет средства для загрузки файлов и библиотек.

10. Перегруженные методы out.println() класса System и их использование для вывода в консоль

System – КЛАСС

out – ПОЛЕ КЛАССА

println() – МЕТОД КЛАССА

С помощью перегрузки метода out.println можно выводить в консоль все примитивные типы данных.

11.Константы в Джава: объявление константы

Для объявления констант в java используется ключевое слово final.
Константе, в отличии от переменной, значение может быть присвоено только один раз.

12.В результате выполнения этой строчки

13.Объявление и использование бестиповых переменных в Джава

Для объявления бестиповых данных используется ключевое слово var.
Компилятор сам подбирает тип переменных, опираясь на присвоенные им значения.
При объявлении такой переменной сразу должна быть инициализация, иначе будет ошибка.

14.Объявление переменных и инициализация типа класс

Для объявления переменной в Java используют следующий синтаксис:
тип данных переменная [= значение], [переменная [= значение], ...] ;
Идём дальше: если нужно объявить больше чем одну переменную указанного типа, допускается применение списка с запятыми:

```
-----------------------------------------------------------------
int a, b, c;         // объявление трёх целых переменных a, b и c
-----------------------------------------------------------------
```

Инициализация полей класса значения по умолчанию

```
-----------------------------------------------------------------
class Up {
   static boolean b;
   static byte by;
   static char c;
   static double d;
   static float f;
   static int i;
   static long l;
   static short s;
   static String st;
}
-----------------------------------------------------------------
```

15.Арифметические операции, операции инкремента и декремента в Джава

«+» операция сложения двух чисел:

```
-----------------------------------------------------------------
int a = 10;
int b = 7;
int c = a + b;  // 17
int d = 4 + b;  // 11
-----------------------------------------------------------------
```

«-» операция вычитания двух чисел:

```
int a = 10;
int b = 7;
int c = a - b;  // 3
int d = 4 - a;  // -6
```

«*» операция умножения двух чисел

```
int a = 10;
int b = 7;
int c = a * b;  // 70
int d = b * 5;  // 35
```

«/» операция деления двух чисел:

```
int a = 20;
int b = 5;
int c = a / b;  // 4
double d = 22.5 / 4.5;  // 5.0
```

«%» получение остатка от деления двух чисел:

```
int a = 33;
int b = 5;
int c = a % b;  // 3
int d = 22 % 4; // 2 (22 - 4*5 = 2)
```

«++» (префиксный инкремент)

Предполагает увеличение переменной на единицу, например, z=++y (вначале значение переменной y увеличивается на 1, а затем ее значение присваивается переменной z)

```
int a = 8;
int b = ++a;
System.out.println(a);  // 9
System.out.println(b);  // 9
```

«++» (постфиксный инкремент)

Также представляет увеличение переменной на единицу, например, z=y++ (вначале значение переменной y присваивается переменной z, а потом значение переменной y увеличивается на 1)

```
int a = 8;
int b = a++;
System.out.println(a);  // 9
System.out.println(b);  // 8
```

«--» (префиксный декремент)
Уменьшение переменной на единицу, например, z=--y (вначале значение переменной y уменьшается на 1, а потом ее значение присваивается переменной z)

```
int a = 8;
int b = --a;
System.out.println(a);  // 7
System.out.println(b);  // 7
```

«--» (постфиксный декремент)
z=y-- (сначала значение переменной y присваивается переменной z, а затем значение переменной y уменьшается на 1)

```
int a = 8;
int b = a--;
System.out.println(a);  // 7
System.out.println(b);  // 8
```

16.В результате выполнения фрагмента программы

17.Арифметические операции, приоритет выполнения операций

«+» операция сложения двух чисел:

```
int a = 10;
int b = 7;
int c = a + b;  // 17
int d = 4 + b;  // 11
```

«-» операция вычитания двух чисел:

```
int a = 10;
int b = 7;
int c = a - b;  // 3
int d = 4 - a;  // -6
```

«*» операция умножения двух чисел

```
int a = 10;
```

```
int b = 7;
int c = a * b;  // 70
int d = b * 5;  // 35
```

«/» операция деления двух чисел:

```
int a = 20;
int b = 5;
int c = a / b;  // 4
double d = 22.5 / 4.5;  // 5.0
```

«%» получение остатка от деления двух чисел:

```
int a = 33;
int b = 5;
int c = a % b;  // 3
int d = 22 % 4; // 2 (22 - 4*5 = 2)
```

Одни операции имеют больший приоритет, чем другие, и поэтому выполняются вначале. Операции в порядке уменьшения приоритета:
++ (постфиксный инкремент), -- (постфиксный декремент)
++ (префиксный инкремент), -- (префиксный декремент)
* (умножение), / (деление), % (остаток от деления)
+ (сложение), - (вычитание)

18.Типы данных в языке Джава, классификация, примеры

boolean: хранит значение true или false

```
boolean isActive = false;
boolean isAlive = true;
```
---
byte: хранит целое число от -128 до 127 и занимает 1 байт

```
byte a = 3;
byte b = 8;
```
---
short: хранит целое число от -32768 до 32767 и занимает 2 байта

```
short a = 3;
short b = 8;
```
---
int: хранит целое число от -2147483648 до 2147483647 и занимает 4 байта
---

int a = 4;
int b = 9;
---
long: хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт
---
long a = 5;
long b = 10;
---
double: хранит число с плавающей точкой от ±4.9*10-324 до ±1.7976931348623157*10308 и занимает 8 байт
---
double x = 8.5;
double y = 2.7;

В качестве разделителя целой и дробной части в дробных литералах используется точка.

float: хранит число с плавающей точкой от -3.4*1038 до 3.4*1038 и занимает 4 байта
---
float x = 8.5F;
float y = 2.7F;
---

19. **Массивы в Джава, объявление и инициализация массивов, длина массива, получение доступа к элементу массива**

а) Объявление:
1) typeData Name [] = new typeData[length]; - стиль Java
2) typeData []Name = new typeData[length]; - стиль из C++
б) Инициализация:
1) int c [] = new int[2];
   c[0] = 1;
   c[1] = 2;
2) int c [] = new int [] {1, 2, 3, 4};
3) int c [] = {1, 2, 3, 4};

20. **Массивы в Джава, как объектные типы данных, контроль доступа за выход за границы массива**

Выход за границы массива
За выходом за границы массивы следит интерпритатор.
Если случился выход за гран. массива, то вылетит исключение ArrayIndexOutOfBoundsException.

21. **Операции над массивами, просмотр элементов массива, поиск по образцу, сортировка массива, сумма элементов массива**

Операции над массивом
а) Поиск - binarySearch
Выполняет поиск с помощью алгоритма бинарного поиска. Имеет перегрузки.
б) Сортировка - sort
в) Сумма всех элементов - Arrays.stream(NameArray).sum()

22. В результате выполнения фрагмента программы

23. Операция конкатенации строк в Джава, ее обозначение и использование и ее использование

В Java присутствует большое колличество решений для работы со строками.
а) Класс StringBuilder
Пример:

```
---------------------------------------------------------------
StringBuilder stringBuilder = new StringBuilder(100);

stringBuilder.append("Baeldung");
stringBuilder.append(" is");
stringBuilder.append(" awesome");

assertEquals("Baeldung is awesome", stringBuilder.toString());
---------------------------------------------------------------
```

б) Оператор сложения "+".
При компиляции символ "+" преобразуется в StringBuilder.append().
Использование "+" в цикле является плохой практикой, т.к каждый раз создаётся новый объёкт String, т.к. объект String явл. неизменяемым в Java.
Пример:

```
---------------------------------------------------------------
String myString = "The " + "quick " + "brown " + "fox...";

assertEquals("The quick brown fox...", myString);
---------------------------------------------------------------
```

в) Строковые методы
В классе String имеется ряд методов для объединения строк.

в.1)Метод String.concat
Пример:

```
---------------------------------------------------------------
String myString = "Both".concat(" fickle")
  .concat(" dwarves")
  .concat(" jinx")
  .concat(" my")
  .concat(" pig")
  .concat(" quiz");

println(myString);
```

в.2) Метод String.format
Метод String.format позволяет преобразовывать различные объекты в строковый шаблон.
В шаблоне находится символ "%" для представлении того, как должны быть расположены в шаблоне различные объекты
Пример:

```
String myString = String.format("%s %s %.2f %s %s, %s...", "I",
  "ate",
  2.5056302,
  "blueberry",
  "pies",
  "oops");

println(myString);
```

в.3) Метод String.Join
В Java 8 и выше можно воспользоваться методом String.Join.
С помощью него мы можем объединить массив строк с помощью общего разделителя.
Пример:

```
String[] strings = {"I'm", "running", "out", "of", "pangrams!"};

String myString = String.join(" ", strings);

println(myString);
```

г) Метод StringJoiner()
Простой в использование класс.
Констурктор принимает разделитель с необязательным префиксом и суффиксом.
Пример:

```
StringJoiner fruitJoiner = new StringJoiner(", ");

fruitJoiner.add("Apples");
fruitJoiner.add("Oranges");
fruitJoiner.add("Bananas");
println(myString);
```

д)Array.toString
Класс Array содержит метод toString, который форматирует массив объектов в

строку.
Данный метод может вывести только строку в квадратных скобках.
Пример:

```
String[] myFavouriteLanguages = {"Java", "JavaScript", "Python"};

String toString = Arrays.toString(myFavouriteLanguages);

println(myString);
```
---

e) Collectors.joining
Метод Collectors.joining позволяет направлять выходные данные потока в одну строку.
Потоки имеют множество возможностей для редактирования строки.
Пример:

```
List<String> awesomeAnimals = Arrays.asList("Shark", "Panda", "Armadillo");

String animalString = awesomeAnimals.stream().collect(Collectors.joining(", "));

println(myString);
```
---

24. Циклы в Джава, цикл с предусловием, цикл с постусловием, пример записи и использование. Условие окончания цикла.

Циклы — это разновидность управляющих конструкций для организации многократного выполнения одного и того же участка кода.
while — цикл с предусловием;
do..while — цикл с постусловием;
for — цикл со счетчиком (цикл для);
for each.. — цикл “для каждого…” — разновидность for для перебора коллекции элементов.
while, do.. while и for можно использовать в качестве безусловных циклов.

Цикл while (с предусловием)
Этот цикл имеет следующую синтаксическую структуру:

```
while (<условие выполнения цикла>) {
   <тело цикла>
}
```

В целом он не сильно отличается от цикла for — цикл while не имеет параметров **<начальное действие>** и **<действие после итерации>**, а содержит лишь условие. Это позволяет выполнять **<тело цикла>** до тех пор, пока выражение в условии возвращает true перед каждой итерацией.

Цикл do…while (с постусловием)
Кроме цикла с предусловием while существует вариант, который выполняет хотя бы одну итерацию, а после этого проверяет условие. Это цикл do…while, который называется циклом с постусловием.

Синтаксис do…while:
---
do {
<тело цикла>
} while (<условие выполнения цикла>);

Сначала отрабатывает действие в **<теле цикла>**, а потом проверяется **<условие выполнения цикла>**. Если оно возвращает true, то цикл выполнит действие повторно.

25. Циклы в Джава, итерационный цикл for(), синтаксис, счетчик цикла, условие окончания цикла, модификация счетчика, пример использования,

Цикл for. Разновидности
а) for(<начальная точка>; <условие выхода>; <операторы счетчика>) {}
Пример:

```
public void printAllElements(String[] stringArray) {
  for(int i = 0; i < stringArray.length; i++) {
     System.out.println(stringArray[i]);
  }
}
```

б) for(<Тип элемента> <Имя переменной, куда будет записан очередной элемент> : <Название массива>) {}
Пример:

```
public void printAllElements(String[] stringArray) {
  for(String s : stringArray) {
     System.out.println(s);
  }
}
```
---

26. Способы объявления массивов в Джава, использование операции new для выделения памяти для элементов массива. Объявление с инициализацией, объявление массива определенного размера без инициализации.

а) Объявление:
1) typeData Name [] = new typeData[length]; - стиль Java
2) typeData []Name = new typeData[length]; - стиль из C++
б) Инициализация:
1) int c [] = new int[2];
```
  c[0] = 1;
  c[1] = 2;
```

2) int c [] = new int [] {1, 2, 3, 4};
3) int c [] = {1, 2, 3, 4};

Тема 2. Реализация ООП в Java.

27.Объявление класса на Джава, пример объявления

Модификатор доступа, слово class и название класса:

```
public class Cat {
   String name;
   int age;
}
```

28.Использования this для доступа к компонентам класса.

Если класс содержит какие то свойства внутри себя, то получать доступ к ним можно с помощью this. Достаточно указать название объекта и через точку добавить название свойства с которым хотим работать. А дальше можно присваивать значение переменной и тд. Пример: Есть класс Animal который содержит свойства Age и Gender. При создании объекта может вызывать конструктор где может быть прописано присваивание значений с помощью this, например:

```
this.Age = 8
this.Gender  = “Male”
```

Говоря проще, слово this это указатель на объект класса с которым мы работаем.

29.Создание или инстанцирование объектов типа класс:

30. Что такое класс в Java?

Класс - это шаблонная конструкция, которая позволяет описать в программе объект, его свойства (атрибуты или поля класса) и поведение (методы класса).

31. Модификатор доступа или видимости в Джава, виды и использование

Модификаторы доступа - ключевые слова, которые регулируют уровни доступа к разным частям программы

а) Private

Модификатор доступа private ограничивает доступ из вне к св-вам и методам

класса.
Данные и методы с таким модификатором нельзя наследовать.
б) Protected
Поля и методы с модификатором доступа protected будут видны в пределах одного пакета и в пределах классов наследников
в) package visible\default
Если у переменной, метода, класса и т.д. нет никакого модификатора, Java по умолчанию присваивает им модификатор default
г) Public
Модификатор public разрешает доступ к полям и методам класса из вне.

32.**Чем отличаются static-метод класса от обычного метода класса:**
Обычные методы привязаны к объектам (экземплярам) класса и могут обращаться к обычным-переменным класса (а также к статическим переменным и методам). Статические же методы привязаны к статическому объекту класса и могут обращаться только к статическим переменным и/или другим статическим методам класса.
Чтобы вызвать обычный метод у класса, сначала нужно создать объект этого класса, а только потом вызвать метод у объекта. Вызвать обычный метод не у объекта, а у класса нельзя.
А чтобы вызвать статический метод, достаточно чтобы просто существовал статический объект класса (который всегда существует после загрузки класса в память). Именно поэтому метод main() — статический. Он привязан к статическому объекту класса, для его вызова не нужно создавать никакие объекты.

33.**Для чего используется оператор new?**
Оператор (операторная функция) new создаёт экземпляр объекта, встроенного или определённого пользователем, имеющего конструктор

34.**Можно ли вызвать static-метод внутри обычного метода?**
Статические методы можно вызывать откуда угодно,из любого места программы. А значит, их можно вызывать и из статических методов, и из обычных. Никаких ограничений тут нет.

35.**Как вызвать обычный метод класса внутри static-метода?**
Для того, чтобы вызывать не static метод внутри static метода, нужно создат объект класса внутри static метода и через него вызвать
не static метод

36.**Для чего используется в Джава ключевое слово this?**
Для доступа и/или обращения к свойствам и компонентам объекта, прописанным в классе данного объекта

37.Объявление и использование методов, объявленных с модификатором public static

38. Синтаксис объявления методов, тип возвращаемого значения, формальные параметры и аргументы

39. Методы с пустым списком параметров

40. Стандартные методы класса сеттеры и геттеры, синтаксис и их назначение?

Геттеры при их вызове возвращают значение указанного в коде свойства или компонента объекта класса.
Сеттеры при их вызове меняют значение указанного в коде свойства или компонента на то, что пользователь передает при вызове в качестве параметра или на значение указанное в самом коде сеттера.

```
String GetName(){ return this.name; }
Void SetName(string name) { this.name = name; }
```

41. Может ли быть поле данных класса объявлено как с модификатором static и final одновременно и что это означает?

Когда мы добавляем к полю\методу ключевое слово static, мы привязываем поле\метод к классу, а не к отдельному объекту этого класса.
Это значит, что для обращения к статическому полю\методу не надо создавать объект класса.
Достаточно написать имя класса и через точку обратиться на прямую к статическим свойствам или методам класса

42. Методы класса конструкторы, синтаксис и назначение

43. Может ли класс иметь в своем составе несколько конструкторов?

Да, может. Например при разном количестве подаваемых параметров при создании объекта класса.

44. Может ли конструктор класса возвращать значение?

Конструктор не может возвращать значения (даже значение void). Если в конструкторе написать возвращение значения с помощью оператора return, то компилятор выдаст ошибку.

Тема 3. Реализация наследования в программах на Джаве

45. *Наследование в Джава. Вид наследования и синтаксис Ключевое слово extends*

46. Что означает перегрузка метода в Java (overload)?

В программировании, перегрузка метода означает использование одинакового имени метода с разными параметрами. Перегрузка методов — это приём программирования, который позволяет разработчику в одном классе для методов с разными параметрами использовать одно и то же имя. В этом случае мы говорим, что метод перегружен.

47. Что означает переопределение метода в Java (override)?

Переопределение метода (англ. Method overriding) — это возможность реализовать метод так, чтобы он имел идентичную сигнатуру с методом класса-предка, но предоставлял иное поведение, не вызывая коллизий при его использовании

48. В чем разница между перегрузкой и переопределением методов, поясните

Перегрузка метода связана с понятием наличия двух или более методов в одном классе с одинаковым именем, но разными аргументами.

void foo(int a)
void foo(int a, float b)

Переопределение метода означает наличие двух методов с одинаковыми аргументами, но разных реализаций. Один из них будет существовать в родительском классе, в то время как другой будет находиться в производном или дочернем классе. @OverrideАннотация, хотя и не является обязательной, может быть полезна для обеспечения надлежащего переопределения метода во время компиляции.

```
class Parent {
   void foo(double d) {
      // do something
   }
}

class Child extends Parent {

   @Override
   void foo(double d){
      // this method is overridden.
   }
}
```

49. ***Абстрактные классы в Джава и абстрактные методы класса***

50. ***Виды наследования в Джава, использование интерфейсов для реализации наследования***

51. ***Что наследуется при реализации наследования в Джава (какие компоненты класса), а что нет?***

52. ***К каким методам и полям базового класса производный класс имеет доступ (даже если базовый класс находится в другом пакете), а каким нет?Область видимости полей и данных из производного класса***

Тема 4. Полиморфизм в Джава. Работа со строками.Интерфейсы.

53. Объявление и инициализация переменных типа String

String имя;
Как и в случае с типами int и double, переменные типа String можно инициализировать сразу при создании. Это, кстати, можно делать вообще со всеми типами в Java.
String name = "Аня", city = "New York", message = "Hello!";

54.Операция конкатенации строк и ее использование

Конкатенация строк – это процесс объединения двух или нескольких строк в одну новую строку. Конкатенация выполняется с помощью оператора +. Символ + также является оператором сложения в математических операциях. Для примера попробуйте объединить две короткие строки:" Sea" + "horse"; Seahorse. Конкатенация объединяет конец одной строки с началом другой строки, не вставляя пробелов.

55. Что означает утверждение, что объект класса String является неизменяемым

Методы могут только создавать и возвращать новые строки, в которых хранится результат операции.

56. При создании объектов строк с помощью класса StringBuffer, например StringBuffer strBuffer = new StringBuffer(str) можно ли использовать операцию конкатенации строк или необходимо использовать методы класса StringBuffer

Нет, так как это разные классы.

57. Объявление и инициализация массива строк. Организация просмотра элементов массива

Общая форма объявления и выделение памяти для одномерного массива строк:
String[] arrayName = **new** String[size];

Инициализация одномерного массива строк точно такая же как инициализация одномерого массива любого другого типа.

```
String[] M = {
   "Sunday",
   "Monday",
   "Tuesday",
   "Wednesday",
   "Thursday",
   "Friday",
   "Saturday"
};
String s;

s = M[2]; // s = "Tuesday"
s = M[4]; // s = "Thursday"
```

58. Понятие и объявление интерфейсов в Джава

Интерфейс в языке программирования Java - это абстрактный тип, который используется для указания поведения, которое должны реализовывать классы.

59. Может ли один класс реализовывать несколько интерфейсов?

Запрет. В одном классе просто запрещается реализовывать несколько интерфейсов, имеющих методы с одинаковыми сигнатурами. Если для какого-то класса требуется комбинация несовместимых интерфейсов, программист должен выбрать другой путь решения проблемы, например, выделить несколько классов, каждый из которых реализует один из необходимых интерфейсов, и использовать их экземпляры совместно.

60. Что входит в состав интерфейса. (какие компоненты может содержать интерфейс)?

С точки зрения программного обеспечения в состав интерфейса входят два компонента: набор процессов ввода-вывода и процесс диалога. Пользователь вычислительной системы взаимодействует с интерфейсом, через интерфейс он посылает входные данные и принимает выходные. Процессы по выполнению заданий вызываются интерфейсом в требуемые моменты времени.
На теоретическом уровне интерфейс имеет три основных компоненты:
Способ общения машины с человеком-оператором
Способ общения человека-оператора с машиной (компьютером)
Способ пользовательского представления интерфейса

61. Может ли интерфейс наследоваться от другого интерфейса?

Интерфейс может наследоваться от другого интерфейса через ключевое слово extends. В этом случае класс должен будет реализовать как методы реализуемого им интерфейса, так и методы его интерфейса-предка.

62. Интерфейс Camparable, назначение, его методы и использование в Джава

63. Какое значение возвращает вызов метода object1.compareTo(object2), который сравнивает 2 объекта obj1 и obj2 в зависимости от объектов?

64. Интерфейсные ссылки и их использование в Джава

Тема 5. Основные принципы и типы исключительных ситуаций.

65. Понятие исключительной ситуации и ее обработка

Java, *исключение* – это специальный объект, описывающий исключительную ситуацию, которая возникла в некоторой части программного кода. Объект представляющий исключение, генерируется в момент возникновения исключительной ситуации. После возникновения критической (исключительной) ситуации исключение перехватывается и обрабатывается. Таким образом, возникает понятие *генерирования исключения*. Генерирование исключения – процесс создания объекта, который описывает данное исключение.

66. В каком случае программа должна использовать оператор throw?

Если метод может породить исключение, которое он сам не обрабатывает, он должен задать это поведение так, чтобы вызывающий его код мог позаботиться об этом исключении.

67. В Java все исключения делятся на два основных типа. Что это за типы и какие виды ошибок ни обрабатывают?

Все исключения в Java делятся на 2 категории — проверяемые (*checked*) и непроверяемые (*unchecked*).

Все исключения, унаследованные от классов RuntimeException и Error, считаются *unchecked-исключениями*, все остальные — *checked-исключениями*.

63. Код ниже вызовет ошибку: Exception <...> java. lang.ArrayIndexOutOfBoundsException: 4: Что она означает?

ArrayIndexOutOfBoundsException – это исключение, появляющееся во время выполнения. Оно возникает тогда, когда мы пытаемся обратиться к элементу массива по отрицательному или превышающему размер массива индексу.

64. Контролируемые исключения (checked)

65. Неконтролируемые исключения (unchecked) и ошибки, которые они обрабатывают

66. Как реализуется принципы ООП в Java при создании исключений?

72. Какой оператор позволяет принудительно выбросить исключение?

Это оператор throw: throw new Exception ();

73. Порядок выполнения операторов при обработке блока блока try...catch

При использовании блока try...catch вначале выполняются все инструкции между операторами try и catch. Если в блоке try вдруг возникает исключение, то обычный порядок выполнения останавливается и переходит к инструкции catch.

Тема 6. Дженерики и использование контейнерных классов в Джава

74. Абстрактный тип данных Stack (стек) в Java

Стек (Stack).

Стековая память отвечает за хранение ссылок на объекты кучи и за хранение

типов значений (также известных в Java как примитивные типы), которые содержат само значение, а не ссылку на объект из кучи.
Кроме того, переменные в стеке имеют определенную видимость, также называемую областью видимости. Используются только объекты из активной области.

75.Универсальные типы или обобщенные типы данных, для чего создаются?

Универсальный тип является одиночным элементом программирования, который используется для выполнения одинаковой функциональности для различных типов данных. При определении универсальных классов или процедур не нужно определять отдельную версию для каждого типа данных, для которых может потребоваться выполнение этой функциональности.

76.Объявление обобщённого класса коллекции с параметризованным методом для обработки массива элементов коллекции на основе цикла for each (определение общего метода для отображения элементов массива)

ArrayList – это реализация динамического использования массива в Java.
ArrayList следует использовать, когда в приоритете доступ по индексу, так как эти операции выполняются за константное время. Добавление в конец списка в среднем тоже выполняется за константное время.

77.Что представляет из себя класс ArrayList и в каком случае используется

ArrayList – это реализация динамического использования массива в Java.
ArrayList следует использовать, когда в приоритете доступ по индексу, так как эти операции выполняются за константное время. Добавление в конец списка в среднем тоже выполняется за константное время.

78.Класс Pattern и его использование

Pattern Class – объект класса Pattern представляет скомпилированное представление регулярного выражения. В классе Pattern публичный конструктор не предусмотрен. Для создания шаблона, вам сперва необходимо вызвать один из представленных публичных статичных методов compile(), который далее произведет возврат объекта класса Pattern.

79. Класс Math и его использование

Класс Math содержит методы для того, чтобы выполнить основные числовые операции, такие как элементарный экспоненциал, логарифм, квадратный корень, и тригонометрические функции. В отличие от некоторых из числовых методов класса StrictMath, все реализации эквивалентных функций класса Math не определяются, чтобы возвратить поразрядное те же самые результаты.

- Класс Math используется Вычисление абсолютных значений (значений по модулю)
- Вычисление значений тригонометрических функций (синусов, косинусов и т.д.)
- Возведение в различные степени
- Извлечение корней различных степеней
- Генерация случайных чисел
- Округления

80. Как вызываются методы класса Math и что при этом происходит?

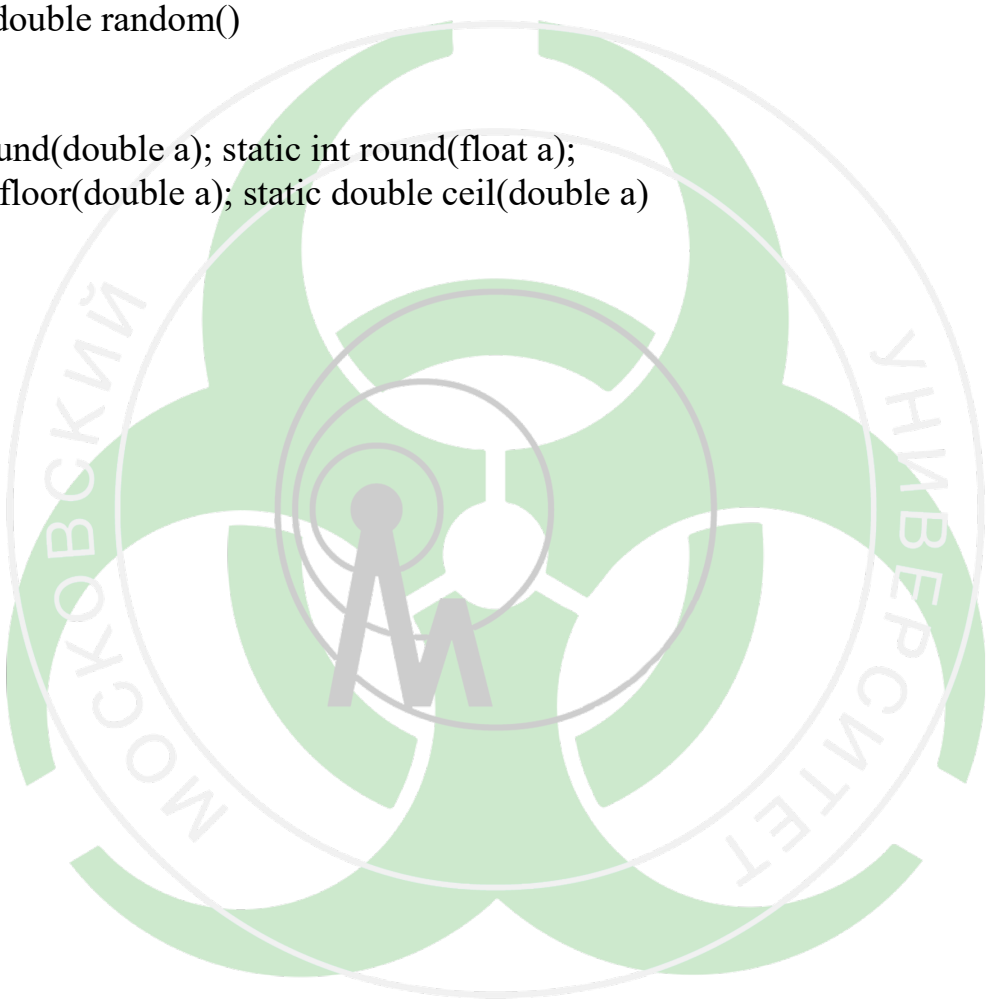
Вычисление значений тригонометрических функций:
static double sin(double a); static double cos(double a); static double cos(double a);
static double cos(double a); static double tan(double a); static double asin(double a)
static double acos(double a); static double atan(double a)

Возведение в степень:
static double pow(double a, double b)

Извлечение корней:
static double sqrt(double a); static double cbrt(double a)

Генерация случайных чисел:
public static double random()

Округление:
static long round(double a); static int round(float a);
static double floor(double a); static double ceil(double a)



Тема 7. Java Core. Дженерики (продолжение) и использование контейнерных классов Java Framework Collection

81. Структура коллекций в Java Collection Framework. Иерархия интефейсов

82. Коллекция HashMap, создание и методы работы с ней

83. Чем является класс LinkedList<E>

Обобщенный класс LinkedList<E> представляет структуру данных в виде связанного

списка. Он наследуется от класса AbstractSequentialList и реализует интерфейсы List, Dequeue и Queue. То есть он соединяет функциональность работы со списком и фукциональность очереди

84. ***Одним из ключевых методов интерфейса Collection является метод Iterator<E> iterator(). Что возвращает это метод?***

85. **Что возвращает метод next()**

Метод next() возвращает следующий (очередной) элемент коллекции.

86. **Что возвращает метод hasNext()**

Метод hasNext() используется, чтобы проверять, есть ли еще элементы. True или False

87. ***Обобщенный класс HashSet класс коллекция, наследует свой функционал от класса AbstractSet, а также реализует интерфейс Set. Что он себя представляет?***

88. ***Обобщенный класс HashMap класс коллекция, которая реализует интерфейс Map для хранения пар ключ-значение. Что он себя представляет?***

Тема 8. Стандартные потоки ввода-вывода. Сериализация.

89. **Стандартные поток ввода-вывода, предоставляемые Java**

Для ввода данных используется класс Scanner из библиотеки пакетов Java.
Для работы с потоком ввода необходимо создать объект класса Scanner, при создании указав, с каким потоком ввода он будет связан. Стандартный поток ввода (клавиатура) в Java представлен объектом — System.in. А стандартный поток вывода (дисплей) — уже знакомым вам объектом System.out. Есть ещё стандартный поток для вывода ошибок — System.err, но работа с ним выходит за рамки нашего курса.

90. **Понятие сериализации, интерфейс Serializable**

Сериализация — это процесс сохранения состояния объекта в последовательность байт.
Serializable - это маркерный интерфейс (не имеет элемента данных и метода). Он используется для “маркировки” классов java, чтобы объекты этих классов могли получить определенные возможности.

91. Какие объекты можно сериализовать?

Сразу надо сказать, что сериализовать можно только те объекты, которые реализуют интерфейс Serializable. Этот интерфейс не определяет никаких методов, просто он служит указателем системе, что объект, реализующий его, может быть сериализован. Сериализация. Класс ObjectOutputStream.

92.Какие методы определяет интерфейс Serializable?

Интерфейс Serializable обеспечивает автоматическую сериализацию, используя инструменты Java Object Serialization. Serializable не объявляет методов: он действует как маркер, сообщая инструментам Object Serialization, что класс Бина является сериализуемым. Пометка класса Serializable означает, что JVM сообщают, что уверены, что класс будет работать с сериализацией по умолчанию.

93.Что означает понятие десериализация?

Десериализация — это обратный процесс: восстановление структур и объектов из сериализованной строки или последовательности байтов.

94. Класс File, определенный в пакете java.io, не работает напрямую с потоками. В чем состоит его задача?

Его задачей является управление информацией о файлах и каталогах.

95. При работе с объектом класса FileOutputStream происходит вызов метода FileOutputStream.write(), что в результате этого происходит?

Главное назначение класса FileOutputStream — запись байтов в файл.