Java API

---

1. Создание, модификация и сравнение объектов класса String. Пул литералов.

String - это один из наиболее широко используемых классов в Java, который находится в пакете java.lang. Это неизменяемый (immutable) и финализированный тип данных, представляющий последовательность символов.

Объект класса String можно создать несколькими способами:

1. Используя последовательность символов в двойных кавычках:

```
String catName = "Barsik";
```

2. С помощью конструкторов:

```
String emptyString = new String();
String dogName = new String("Jack");
String copyOfCatName = new String (catName);
String copyOfDogName = dogName;
```

Конструкторы могут формировать объект строки с помощью массива символов или массива байтов:

```
char[] arrayOfChars = {'M', 'u', 'r', 'z', 'i', 'k'};
String newCatName = new String (arrayOfChars);
```

Строки являются неизменными, поэтому при попытке изменить объект String создается новый объект String. Но ссылку на объект можно изменить так, чтобы она указывала на другой объект.

Методы сравнения строк:

boolean equals(Object obj)	сравнение строк
boolean equalsIgnoreCase(String str2)	сравнение строк без учета регистра символов
int compareTo(String str2)	лексиграфическое сравнение строк, вернет 0, если строки равны
int compareToIgnoreCase(String str)	лексиграфическое сравнение строк без учета регистра символов
boolean contentEquals(CharSequence cs)	сравнивает строку с объектом типа CharSequence
boolean contentEquals(StringBuffer sb)	сравнивает строку с объектом типа StringBuffer

Сравнение объектов класса String происходит при помощи метода equals:

```
System.out.println(catName.equals(copyOfCatName)); //true
```

Строки нельзя сравнивать оператором ==, хотя он и работает со строками. Дело в том, что оператор == сравнивает адреса объектов в памяти. То есть если сравниваемые переменные String указывают на один и тот же объект в памяти, то результат сравнения будет true. В противном случае - false, даже если строки будут равны посимвольно:

```
System.out.println(catName == copyOfCatName); //false!
```

Пул литералов - это коллекция ссылок на строковые объекты в Java Heap. Когда мы используем двойные кавычки для создания строки, сначала в пуле ищется строка с таким же значением. Если такая строка находится, то просто возвращается ссылка на эту строку, иначе создается новая строка в пуле, а затем возвращается ссылка:

```
String cat1 = "Tom";
String cat2 = "Tom";
System.out.println(cat1 == cat2); //true, так как ссылки cat1 и cat2 указывают на одну и
                                  ту же строку в пуле
```

Мы можем "заставить" класс String создать новый объект, независимо от того, есть строка с таким значением в пуле или нет:

```
String cat3 = new String ("Tom");
System.out.println(cat1 == cat3); //false, так как в пуле была создана еще одна строка
                                    с содержимым "Tom".
```

И наоборот: мы можем "заставить" класс String проверять наличие строки в пуле, даже если используется new при создании. Для этого существует метод intern:

```
String cat4 = new String ("Tom").intern();
System.out.println(cat1 == cat4); //true, так как новый объект в пуле не был создан, то
                                    есть ссылки cat1 и cat4 указывают на одну
                                    область памяти
```

Пул строк возможен благодаря неизменяемости строк. Он помогает экономить большой объем памяти, но с другой стороны создание строки занимает больше времени.

---

2. Перечислите методы класса Collections. Укажите назначение этих методов.

Collections - это класс состоящий из статических методов, осуществляющих различные служебные операции над коллекциями. К ним можно обращаться так:

```
Collections.method(Collection);
```

Для работы с любой коллекцией:

frequency(Collection, Object)	возвращает кол-во вхождений элемента в коллекции
disjoint(Collection, Collection)	true, если в коллекциях нет общих элементов
addAll(Collection, T[])	добавляет все элементы массива в коллекцию
min(Collection)	минимальный элемент коллекции
max(Collection)	максимальный элемент коллекции

Для работы со списками:

fill(List, Object)	заменяет каждый элемент в списке значением
sort(List)	сортирует слиянием за O(n*log n)
reverse(List)	изменяет порядок всех элементов
rotate(List, int)	передвигает все элементы на заданный диапазон
binarySearch(List, Object)	ищет элемент (бинарный поиск)
shuffle(List)	перемешивает элементы в случайном порядке
copy(List dest, List src)	копирует один список в другой
replaceAll(List, Object old, Object new)	заменяет все вхождения одного значения на другое
swap(List, int, int)	меняет местами элементы на указанных позициях
indexOfSubList(List src, List trg)	индекс первого вхождения списка trg в список src
lastIndexOfSubList(List src, List trg)	индекс последнего вхождения списка trg в список src

Создание копий коллекции:

newSetFromMap(Map)	создает Set из Map
unmodifiableCollection(Collection)	создает неизменяемую копию коллекции
synchronizedCollection(Collection)	создает потокобезопасную копию коллекции
asLifoQueue(Deque)	создает очередь last in first out из Deque
<T> Set<T> singleton(T o)	создает неизменяемый Set с заданным объектом (только с ним)

---

3. Класс Object, методы класса Object.

Все классы в Java являются производными класса java.lang.Object, реализуя парадигму Java "все есть объект" (кроме примитивных типов). То есть объект Object может ссылаться на объект любого другого типа, а любому объекту доступны методы класса Object:

Object clone()	создает поверхностную копию вызывающего объекта

boolean equals(Object)	сравнивает содержимое двух объектов, должен быть переопределен для корректной работы с пользовательскими классами
Class getClass()	возвращает класс объекта - объект типа Class, который является набором метаданных класса
void finalize()	вызывается сборщиком мусора автоматически перед удалением объекта; может служить рекомендацией сборщику мусора о начале работы, однако может быть проигнорирована
int hashCode()	возвращает хэш-код объекта - integer, которые является числовым представлением объекта
String toString()	возвращает представление объекта в виде строки, должен быть переопределен для корректной работы с пользовательскими классами

Кроме того, именно в Object определены методы для работы с потоками:

void wait()	поток переходит в режим ожидания
void notify()	просыпается один поток, который ждет на "мониторе" данного объекта
void notifyAll()	просыпаются все потоки, которые ждут на "мониторе" данного объекта

---

4. Методы класса String.

1. char charAt(int index)
   возвращает символ по указанному индексу
2. int compareTo(Object o)
   сравнивает данную строку с другим объектом
3. int compareToIgnoreCase(String str)
   сравнивает две строки лексически, игнорируя регистр символов
4. String concat(String str)
   объединяет указанную строку с данной строкой, путем добавления ее в конец
5. boolean contentEquals(StringBuffer sb)
   возвращает true, если эта строка представляет собой ту же последовательность символов, которая указана в StringBuffer
6. static String copyValueOf(char[] data)
   возвращает строку, которая представляет собой последовательность символов в заданном массиве
7. boolean endsWith(String suffix)
   проверяет, заканчивается ли эта строка указанным окончанием
8. boolean startsWith(String prefix)
   проверяет, начинается ли эта строка с заданного префикса
9. boolean equals(Object anObject)
   сравнивает данную строку с указанным объектом
10. boolean equalsIgnoreCase(String anotherString)
    сравнивает данную строку с другой строкой, игнорируя регистр символов
11. byte[] getBytes(String charsetName)
    кодирует эту строку в последовательность байтов, сохраняя результат в новый массив байтов
12. void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
    копирует символы из строки в массив символов назначения
13. char[] toCharArray()
    преобразует эту строку в новый массив символов
14. int hashCode()
    возвращает хэш-код строки
15. int indexOf(int ch)
    возвращает индекс первого вхождения указанного символа в строке
16. int indexOf(String str)
    возвращает индекс первого вхождения указанной подстроки в данной строке
17. int lastIndexOf(int ch)

возвращает индекс последнего вхождения указанного символа в строке
18. int lastIndexOf(String str)
возвращает индекс последнего вхождения указанной подстроки в данной строке
19. String intern()
возвращает каноническое представление для строкового объекта
20. int length()
возвращает длину строки
21. boolean matches(String regex)
сообщает, соответствует ли эта строка заданному регулярному выражению
22. String replace(char oldChar, char newChar)
возвращает новую строку после замены всех вхождения oldChar на newChar
23. String replaceAll(String regex, String replacement)
заменяет каждую подстроку, соответствующую заданному регулярному выражению, указанной строкой
24. String replaceFirst(String regex, String replacement)
заменяет первые подстроки, соответствующие заданному регулярному выражению, указанной строкой
25. String[] split(String regex)
разделяет строку по регулярному выражению
26. CharSequence subSequence(int beginIndex, int endIndex)
возвращает новую последовательность символов, которая является подстрокой строки
27. String substring(int beginIndex, int endIndex)
возвращает новую строку, которая является подстрокой строки
28. String toLowerCase()
преобразует все символы в строке в нижний регистр
29. String toUpperCase()
преобразует все символы в строке в верхний регистр
30. String toString()
строка возвращает себя
31. String trim()
возвращает копию строки, обрезая начальные и конечные пробелы
32. String valueOf(primitive data type x)
возвращает строковое представление переданного аргумента

---

5. Работа с объектами типа StringBuilder и StringBuffer.

Строки в Java являются неизменяемыми объектами, но часто необходимо получить такой строковый объект, который можно будет изменять. Для этого используются классы StringBuilder и StringBuffer, объекты которых являются изменяемыми. Единственным отличием StringBuilder от StringBuffer является потокобезопасность StringBuffer. В однопоточном использовании StringBuilder практически всегда значительно быстрее, чем StringBuffer. Для этих классов не переопределены методы equals() и hashCode(), то есть сравнить содержимое двух объектов невозможно, а хэш-коды всех объектов этого типа вычисляются так же, как и для класса Object.

Конструкторы StringBuilder:
StringBuilder(String str) – создает StringBuilder, значение которого устанавливается в передаваемую строку, плюс дополнительные 16 пустых элементов в конце строки

StringBuilder(CharSequence charSeq) – создает StringBuilder, содержащий те же самые символы, что в CharSequence, плюс дополнительные 16 пустых элементов, конечных CharSequence

StringBuilder(int length) – создает пустой StringBuilder с указанной начальной вместимостью

StringBuilder() – создает пустой StringBuilder из 16 пустых элементов

Чтение и изменение объекта StringBuilder:

int length() – возвращает количество символов в строке

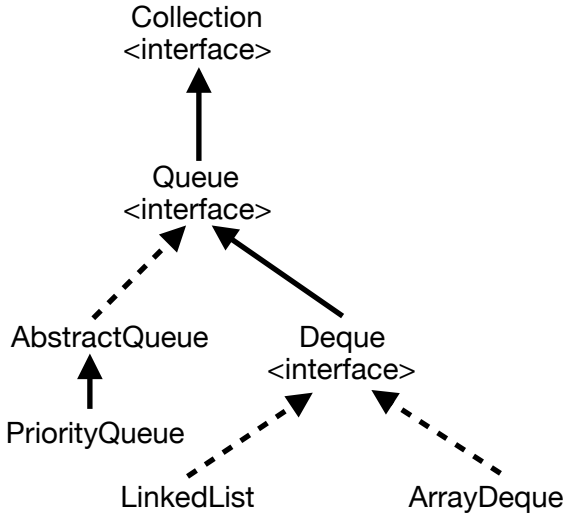
char charAt(int index) – возвращает символьное значение, расположенное на месте index

void setCharAt(int index, char ch) – символ, расположенный на месте index, заменяется символом ch

CharSequence subSequence(int start, int end) – возвращает новую подстроку

---

6. Очереди (Queue, Deque). Конкретные классы очередей.



Очередь (Queue) - это очередь, обычно (но необязательно) строится по принципу FIFO (First-In-First-Out), соответственно извлечение элемента осуществляется с начала очереди, а вставка элемента в конец очереди. Этот принцип нарушает, к примеру, приоритетная очередь (PriorityQueue). Очередь предназначена для хранения элементов перед их обработкой. Кроме базовых методов Collection, очередь предоставляет дополнительные методы по добавлению, извлечению и проверке элементов.

Методы интерфейса Queue:

boolean offer(E o)	добавляет в конец очереди новый элемент, возвращает true, если добавление прошло успешно
E peek()	возвращает первый элемент очереди
E poll()	возвращает и удаляет первый элемент очереди
E element()	возвращает головной элемент очереди
E remove()	возвращает и удаляет головной элемент очереди

PriorityQueue – это класс очереди с приоритетами. По умолчанию очередь с приоритетами размещает элементы согласно естественному порядку сортировки используя Comparable. Элементу с наименьшим значением присваивается наибольший приоритет. Если несколько элементов имеют одинаковый наивысший элемент – связь определяется произвольно. Также можно указать специальный порядок размещения, используя Comparator. Конструктор PriorityQueue() создает очередь с начальной емкостью 11, а элементы размещаются согласно естественному порядку сортировки. Также есть конструкторы, в которых можно задать свою емкость и свой порядок сортировки. При добавлении элементов емкость растет автоматически. Добавление, поиск и извлечение элементов выполняется за O(logn).

Интерфейс Deque расширяет Queue - это двусторонняя очередь, может строиться по принципам FIFO и LIFO. В этой очереди элементы можно добавлять как в начало, так и в конец, а также брать элементы тоже можно и из начала, и из конца очереди.

Методы интерфейса Deque аналогичный методам Queue, но позволяют "работать" с двух сторон:

void addFirst(E e);	аналогичен offerFirst, но выбрасывает exception, если вставка не удалась
void addLast(E e);	
boolean offerFirst(E e);	
boolean offerLast(E e);	
E peekFirst();	
E peekLast();	
E pollFirst();	
E pollLast();	
E removeFirst();	
E removeLast();	
E getFirst();	аналогичен peekFirst, но выбрасывает NoSuchElementException, если очередь пустая
E getLast();	

ArrayDeque - реализация интерфейса Deque переменного размера. Элементы в ArrayDeque расположены линейно и связаны с двумя соседями в обе стороны. Емкость по умолчанию - 16, но при создании можно задать свое значение емкости. Извлечение любого элемента выполняется за O(n), а концевых элементов за O(1).

LinkedList - двусвязный список, реализация интерфейсов List и Deque, однако это "больше" список, чем очередь. Но так как LinkedList позволяет добавлять элементы в начало и конец списка за константное время, это хорошо подходит для реализации интерфейса Deque.

---

7. Переопределение метода toString();

Метод toString в Java используется для предоставления ясной и достаточной информации об объекте (Object) в удобном для человека виде. Правильное переопределение метода toString может помочь в ведении журнала работы и в отладке Java программы, предоставляя ценную и важную информацию. Поскольку toString() определен в классе java.lang.Object и его реализация по умолчанию не предоставляет много информации, всегда лучшей практикой является переопределение данного метода в производном классе. По умолчанию реализация toString создает вывод в виде:

package.class@hashCode

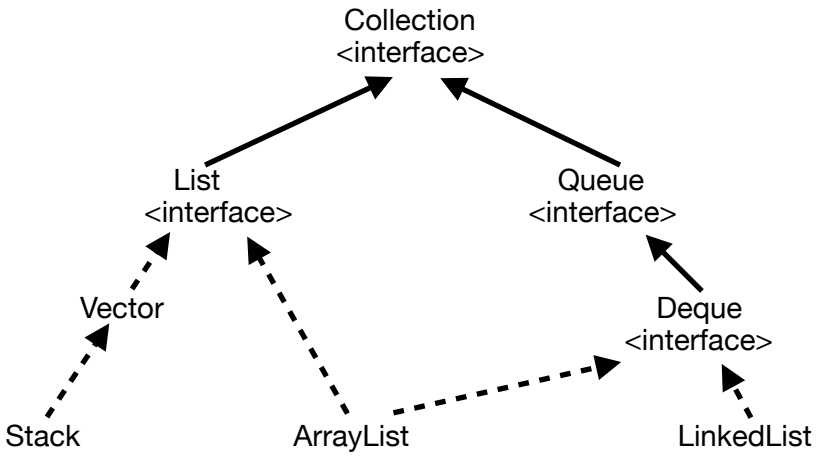
```
public class User {
    private String name;
    private String surname;
    private int birthYear;

    public User(String name, String surname, int birthYear) {
        this.name = name;
        this.surname = surname;
        this.birthYear = birthYear;
    }

    public int getBirthYear() {
        return birthYear;
    }

    @Override
    public String toString() {
        return this.name + " " + this.surname + ", " + getBirthYear() + " г.р.";
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        User user1 = new User("Petya", "Lee", 1988);
        System.out.println(user1.toString());
    }
}
```

/*Output:
Petya Lee, 1988 г.р.

---

8. Основные характеристики списков (List). Конкретные классы списков.



Список (List) - упорядоченная коллекция, в которой могут содержаться повторяющиеся элементы. В списке сохраняется последовательность добавления элементов, поэтому доступно обращение к элементу по индексу.
Особенности:

- работа с элементами, основанная на их позиции в списке (на их индексе): get, set, add, addAll, remove
- обеспечение возможности поиска элемента и возвращение его индекса в списке: indexOf, lastIndexOf
- обеспечение получения подписков, которые являются представление изначального списка
- возможность двустороннего обхода списка, выполнения вставки и замещения элементов

Методы:

E get(int index);	возвращает объект, находящийся в позиции index
E set(int index, E element);	заменяет элемент, находящийся в позиции index объектом element
boolean add(E element);	добавляет элемент в список
boolean addAll(Collection c)	добавляет все элементы коллекции в список
E remove(int index);	удаляет элемент, находящийся на позиции index
void clear()	удаляет все элементы из списка
int indexOf(Object o);	возвращает индекс первого появления элемента
int lastIndexOf(Object o);	возвращает индекс последнего появления элемента
List<E> subList(int from, int to);	возвращает новый список, представляющий собой часть данного (с from до to-1 включительно)

Реализации интерфейса List:

LinkedList - двунаправленный список, то есть каждый элемент списка содержит указатели на предыдущий и следующий элемент в списке. Итератор поддерживает обход в обе стороны. Реализует методы получения, удаления и вставки в любое место списка, при этом быстрое добавление и и удаление элементов является преимуществом структуры.

Позволяет добавлять любые элементы, включая null. Поиск элементов выполняется за O(n), вставка за O(1). Рекомендуется использовать, если необходимо часто вставлять и удалять элементы.

ArrayList - списочный массив. Массивы в Java имеют фиксированную длину, которая не может быть изменена после создания массива. ArrayList имеет возможность менять свой размер после создания. Размер по умолчанию - 10, который каждый раз при заполнении увеличивается в 1.5 раза. Поиск элементов выполняется за O(1), вставка и удаление элемента в конце массива выполняется за O(1), а вставка и удаление из произвольного места за 0(n). В ArrayList нет дополнительных расходов по памяти на хранение связок между элементами.

Vector - аналогичен ArrayList, но потокобезопасный. Работает медленнее, чем ArrayList.

Stack - коллекция, объединяющая элементы в стек. Позволяет создавать очередь типа LIFO (Last-In-First-Out). "Взять" можно только тот элемент, который был добавлен последним. Является потокобезопасным. Методы:

E peek()	возвращает верхний элемент
E pop()	возвращает и удаляет верхний элемент
E push(E item)	добавляет элемент в вершину стека
int search(Object o)	ищет элемент в стеке, возвращая количество операций pop, которые требуются для того, чтобы перевести элемент в вершину стека. Если элемент не найдет, возвращает -1