Билет 2

# 1. Назовите основные методы класса File.

Класс File имеет ряд методов, которые позволяют управлять файлами и каталогами.

* **boolean createNewFile()**: создает новый файл по пути, который передан в конструктор. В случае удачного создания возвращает true, иначе false
* **boolean delete():** удаляет каталог или файл по пути, который передан в конструктор. При удачном удалении возвращает true.
* **boolean exists():** проверяет, существует ли по указанному в конструкторе пути файл или каталог. И если файл или каталог существует, то возвращает true, иначе возвращает false
* **boolean mkdir()**: создает новый каталог и при удачном создании возвращает значение true
* **boolean renameTo(File dest)**: переименовывает файл или каталог
* **String getAbsolutePath()**: возвращает абсолютный путь для пути, переданного в конструктор объекта
* **String getName()**: возвращает краткое имя файла или каталога
* **String getParent()**: возвращает имя родительского каталога
* **boolean isDirectory()**: возвращает значение true, если по указанному пути располагается каталог
* **boolean isFile()**: возвращает значение true, если по указанному пути находится файл
* **boolean isHidden()**: возвращает значение true, если каталог или файл являются скрытыми

# 2. Расскажите про реализацию интерфейса Map и Виды MAP.

|  |  |
| --- | --- |
| Интерфейс **Map**соотносит уникальные ключи со значениями. Ключ — это объект, который вы используете для последующего извлечения данных. Задавая ключ и значение, вы можете помещать значения в объект карты. После того как это значение сохранено, вы можете получить его по ключу. Интерфейс **Map**— это обобщенный интерфейс, объявленный так, как показано ниже.  interface Мар<К, V>  Здесь К указывает тип ключей, а V — тип хранимых значений. |  |

**Виды МАР**

* **HashMap**— основан на хэш-таблицах, реализует интерфейс Map (что подразумевает хранение данных в виде пар ключ/значение). Ключи и значения могут быть любых типов, в том числе и null. Данная реализация не дает гарантий относительно порядка элементов с течением времени.
* **LinkedHashMap**-  расширяет класс **HashMap**. Он создает связный список элементов в карте, расположенных в том порядке, в котором они вставлялись. Это позволяет организовать перебор карты в порядке вставки. То есть, когда происходит итерация по коллекционному представлению объекта класса LinkedHashMap, элементы будут возвращаться в том порядке, в котором они вставлялись. Вы также можете создать объект класса LinkedHashMap, возвращающий свои элементы в том порядке, в котором к ним в последний раз осуществлялся доступ.
* **TreeMap**- расширяет класс **AbstractMap**и реализует интерфейс **NavigatebleMap**. Он создает коллекцию, которая для хранения элементов применяет *дерево*. *Объекты сохраняются в отсортированном порядке по возрастанию*. Время доступа и извлечения элементов достаточно мало, что делает класс TreeMap блестящим выбором для хранения больших объемов отсортированной информации, которая должна быть быстро найдена. **WeakHashMap**- коллекция, использующая слабые ссылки для ключей (а не значений). Слабая ссылка*(англ. weak reference)* — специфический вид ссылок на динамически создаваемые объекты в системах со сборкой мусора. Отличается от обычных ссылок тем, что не учитывается сборщиком мусора при выявлении объектов, подлежащих удалению. Ссылки, не являющиеся слабыми, также иногда именуют «сильными».