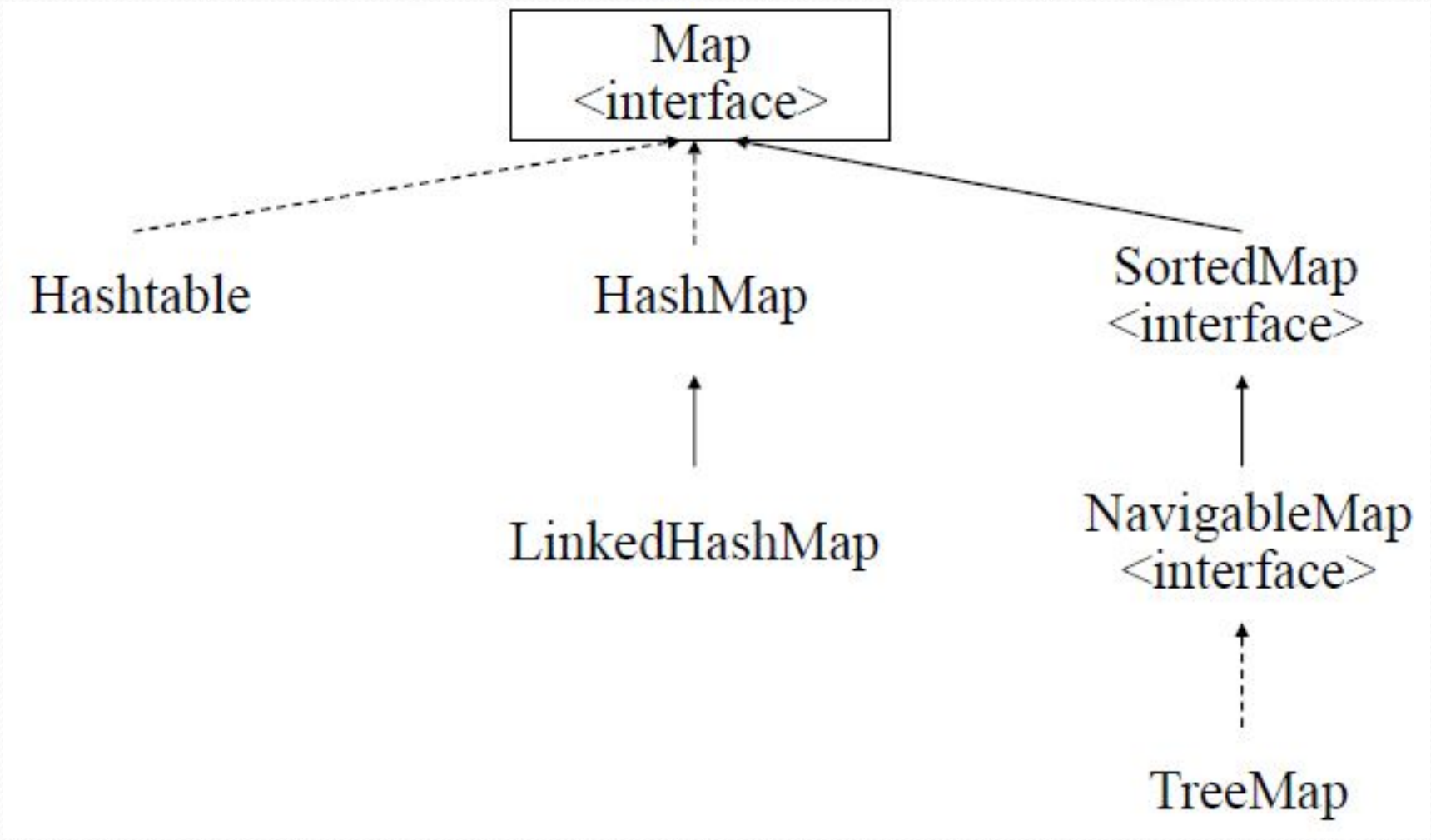
Программирование на Java
Тема 5.3 Коллекции

Интерфейс Map



Интерфейс Map

- Интерфейс Map описывает коллекцию, которую называют ассоциативный массив (словарь, карта отображений). Коллекция Map — это набор пар объектов: ключ и значение, причем ключи являются уникальными. Уникальность ключей определяет реализация метода equals(…).
- В любой момент можно получить элемент-значение по заданному ключу.
- Поиск объекта (значения) облегчается по сравнению с множествами за счет того, что его можно найти по его уникальному ключу. Если элемент с указанным ключом отсутствует в карте, то возвращается значение **null**.
- Для корректной работы с картами необходимо переопределить методы equals(…) и hashCode() .

- Классы, реализующие интерфейс Map :
- **HashMap** – хэш-таблица, в которой ключи отсортированы относительно значений их хэш-кодов.
- Эффективность работы HashMap зависит от того, насколько эффективно реализован метод hashCode(). HashMap может принимать в качестве ключа null, но такой ключ может быть только один, значений null может быть сколько угодно.
- **Hashtable** синхронизированная версия HashMap, так как методы у Hashtable синхронизированы, то Hashtable работает медленнее чем HashMap. Null не может быть использован в качестве ключа Hashtable.
- **LinkedHashMap** – связный список, который хранит элементы в порядке вставки.
- **TreeMap** – дерево, где ключи расположены в виде дерева поиска в порядке сортировки.
- Порядок сортировки может задаваться реализацией интерфейсов Comparator и Comparable.
- Реализация Comparator передается в конструктор TreeMap, Comparable используется при добавлении элемента в карту.

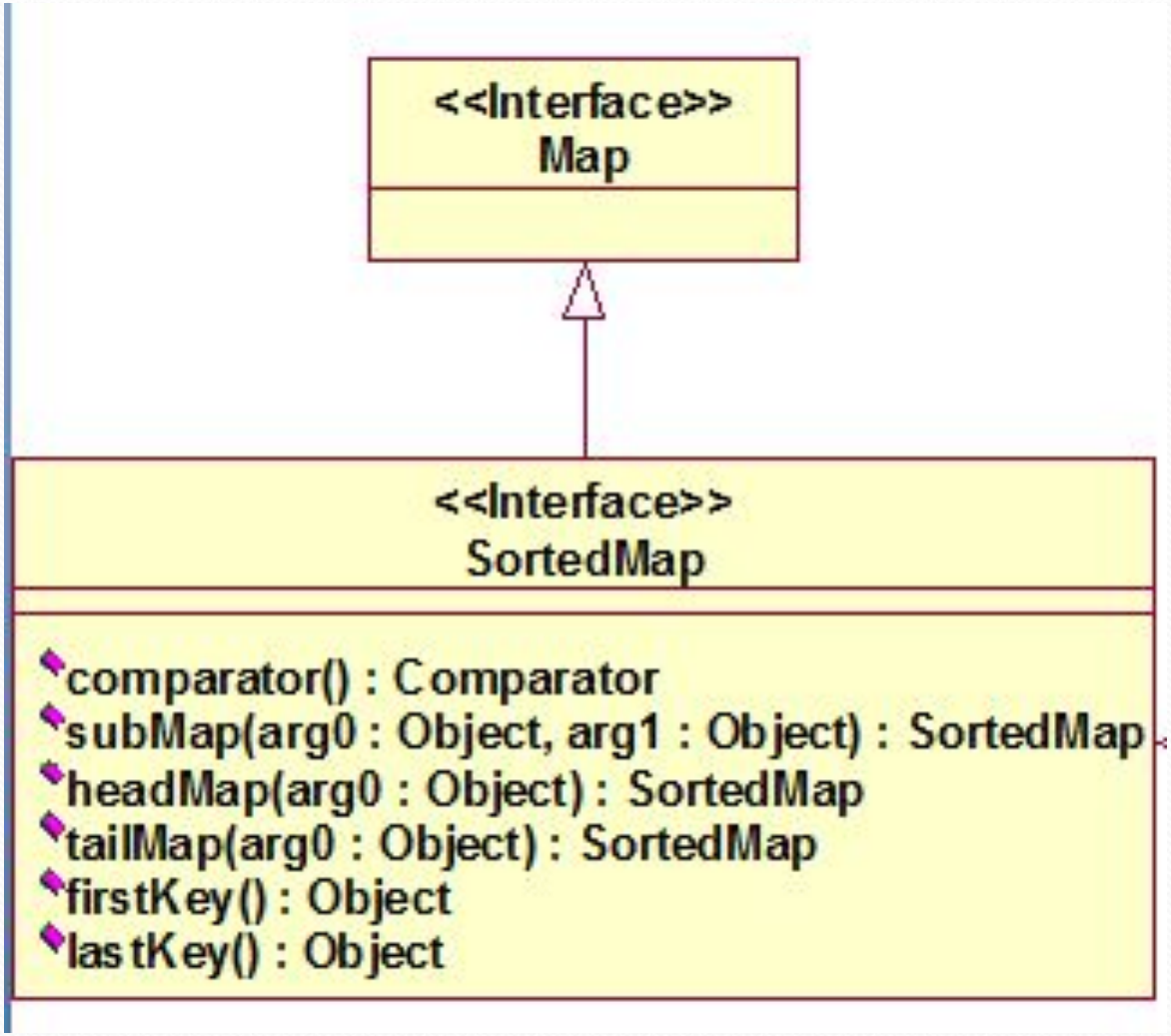
Интерфейс Map

```
public interface Map<K,V>
    { V put(K key, V value);
      V get(Object key);
      V remove(Object key);
      boolean containsKey(Object key);
      boolean containsValue(Object value);
      int size();
      boolean isEmpty();
      void putAll(Map<? extends K, ? extends V> m);
      void clear();
      public Set<K> keySet();
      public Collection<V> values();
      public Set<Map.Entry<K,V>> entrySet();
      public interface Entry {
      K getKey();
      V getValue();
      V setValue(V value);
}
```

Интерфейс Map

- Интерфейс Map — это обобщенный интерфейс, объявленный как : interface Map<K, V>, здесь K указывает тип ключей, а V — тип хранимых значений.
- Интерфейс **Map** содержит методы:
  - *size()* — возвращает количество элементов (пар) в массиве;
  - *containsKey*(Object key) — проверяет, существует ли в массиве элемент с ключом key;
  - *containsValue*(Object value) — проверяет, существует ли в массиве элемент со значением value;
  - *get*(Object key) — возвращает значение, соответствующее ключу key;
  - *put*(Object key, Object value) — добавляет в массив элемент с ключом key и значением value. Если элемент с таким ключом уже существует в массиве, то его значение просто изменяется;
  - *values()* — возвращает значения всех элементов массива в виде коллекции (т.е. возвращаемый результат имеет типCollection);
  - *remove(Object key)* — удаляет элемент с ключом key, возвращая значение этого элемента (если он есть) и **null**, если такого элемента не было;
  - *clear()* — очищает массив;
  - *isEmpty()* — проверяет, не пуст ли массив.
- Каждый элемент ассоциативного массива, описываемого интерфейсом Map, имеет интерфейсный тип Map.Entry, который предоставляет три основных метода:
  - *getKey()* — возвращает ключ элемента;
  - *getValue()* — возвращает значение элемента;
  - *setValue(Object value)* — меняет значение элемента.
- Метод entrySet(), определенный в интерфейсе Map, позволят получить все элементы ассоциативного массива в виде множества объектов типа Map.Entry.

Интерфейс SortedMap



Интерфейс SortedMap

- **SortedMap** — наследник интерфейса Map, описывает ассоциативный массив, элементы которого упорядочены по ключам.
- Методы, предоставляемые этим интерфейсом: firstKey(), lastKey(), subMap(Object fromKey, Object toKey),headMap(Object toKey), tailMap(Object fromKey) аналогичны методам интерфейса SortedSet. Данный интерфейс реализуется, например, в классе **TreeMap**. Один из конструкторов этого класса принимает объект типа Comparator, посредством которого можно задать свой собственный порядок сортировки.

Интерфейс Map и класс HashMap

- **Пример .**

```
import java.util.HashMap;
import java.util.Set;

class Auto {
    private double timeTo100KM;
    private double volumeMotor;
    private int maxSpeed;

    public double getMotor()
    {
        return volumeMotor;
    }
    Auto(double time, double motor, int maxSpeed)
    {
        this.timeTo100KM=time;
        this.volumeMotor=motor;
        this.maxSpeed=maxSpeed;
    }
    public String toString()
    {
        return "<time to 100km: "+timeTo100KM+"sec; motor: "+volumeMotor
        +" l; max speed: "+maxSpeed+"km. >";
    }
}
```

Интерфейс Map и класс HashMap

- **Пример .**

```
class AutoManual {

    private HashMap<String, Auto> guide = new HashMap<String, Auto>();

    public void add(String model, Auto auto)
    {
        guide.put(model, auto);
    }
    public void getModelByMotor(double motor)
    {
        Set<String> autoes = guide.keySet();
        for(String k:autoes)
        {
            if(guide.get(k).getMotor()==motor)
            {
                System.out.println(k);
            }
        }
    }
    public Auto get(String model)
    {
        return guide.get(model);
    }
}
```

Интерфейс Map и класс HashMap

- **Пример .**

```
public class Main2 {
 public static void main(String[] args)
     {
      AutoManual am = new AutoManual();
      am.add("BMW", new Auto(new Double(4.5),new Double(2.0),272));
      am.add("Audi", new Auto(new Double(4.0),new Double(2.0),253));
      am.add("Kopeyka", new Auto(new Double(25.0),new Double(0.5),100));
      Auto auto = am.get("Audi");
      System.out.println("Audi "+auto);
      am.getModelByMotor(new Double(2.0));
     }
}
```

```
Audi <time to 100km: 4.0sec; motor: 2.0 l; max speed: 253km. >
Audi
BMW
```

Контрольные вопросы

1. **Характеристика и примеры использования классов-коллекций и интерфейсов-коллекций для организации списков.**
2. **Интерфейсы Iterator и ListIterator : назначение, примеры использования.**
3. **Интерфейсы Comparable и Comparator: назначение, примеры использования.**
4. **Характеристика и примеры использования классов-коллекций и интерфейсов-коллекций для организации множеств.**
5. **Характеристика и примеры использования классов-коллекций и интерфейсов-коллекций для организации ассоциативных массивов (карт, отображений).**