/ JAVA DB SCHOOL

Generics, Collections

/ Generics

Generics

- Elements with types *to-be-specified-later*
- Instantiated when needed for specific types provided as parameters
- Used with classes, interfaces and methods
- Example of generics class:

```
package java.lang;
public interface Comparable<T> {
      public int compareTo(T o);
 }
```

Generics

- T is a generic format type which can be replaced with an actual type (Object if not set)

- Another example:

```
ArrayList<String> myList = new ArrayList<String> ();
```

vs

```
ArrayList<Double> myList = new ArrayList<Double> ();
```

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

```
public boolean add(E e)
public boolean remove(Object o)
```

Generics

- A generic can have more than one generic parameters: two, three…

```
Class HashMap<K,V>
Interface Map<K,V>
```

https://docs.oracle.com/javase/8/docs/api/java/util/Map.html

Generic Method

```
public class GenericsDemo {
    public static void main(String[] args) {
        Integer[] myIntegerList = { 1, 2, 3, 4, 5 };
        String[] myStringList = { "Hello", "World" };
        GenericsDemo.<Integer>print(myIntegerList);
        GenericsDemo.<String>print(myStringList);
    }

    public static<T> void print(T[] myList) {
        for (int i = 0; i < myList.length; i++) {
            System.out.println(myList[i] + " ");
        }
    }
}
```

Wildcards

- Extends, super

`<? extends Number>` - Number sau un subtip al lui `Number`

`<? extends Object>` - Object sau un subtip al lui Object (orice obiect)

`<? super T>` - T sau un supertip al lui T

Wildcards Example

```
public class Test {
    public static void listAutomobiles(List<? extends Automobile> myList) {
        for (Automobile a : myList)
            System.out.println(a.getType());
    }

    public static void main(String[] args) {
        List<Automobile> myList = new ArrayList<Automobile>();
        myList.add(new Dacia());
        myList.add(new BMW());
        myList.add(new Automobile());
        listAutomobiles(myList);
    }
}
```

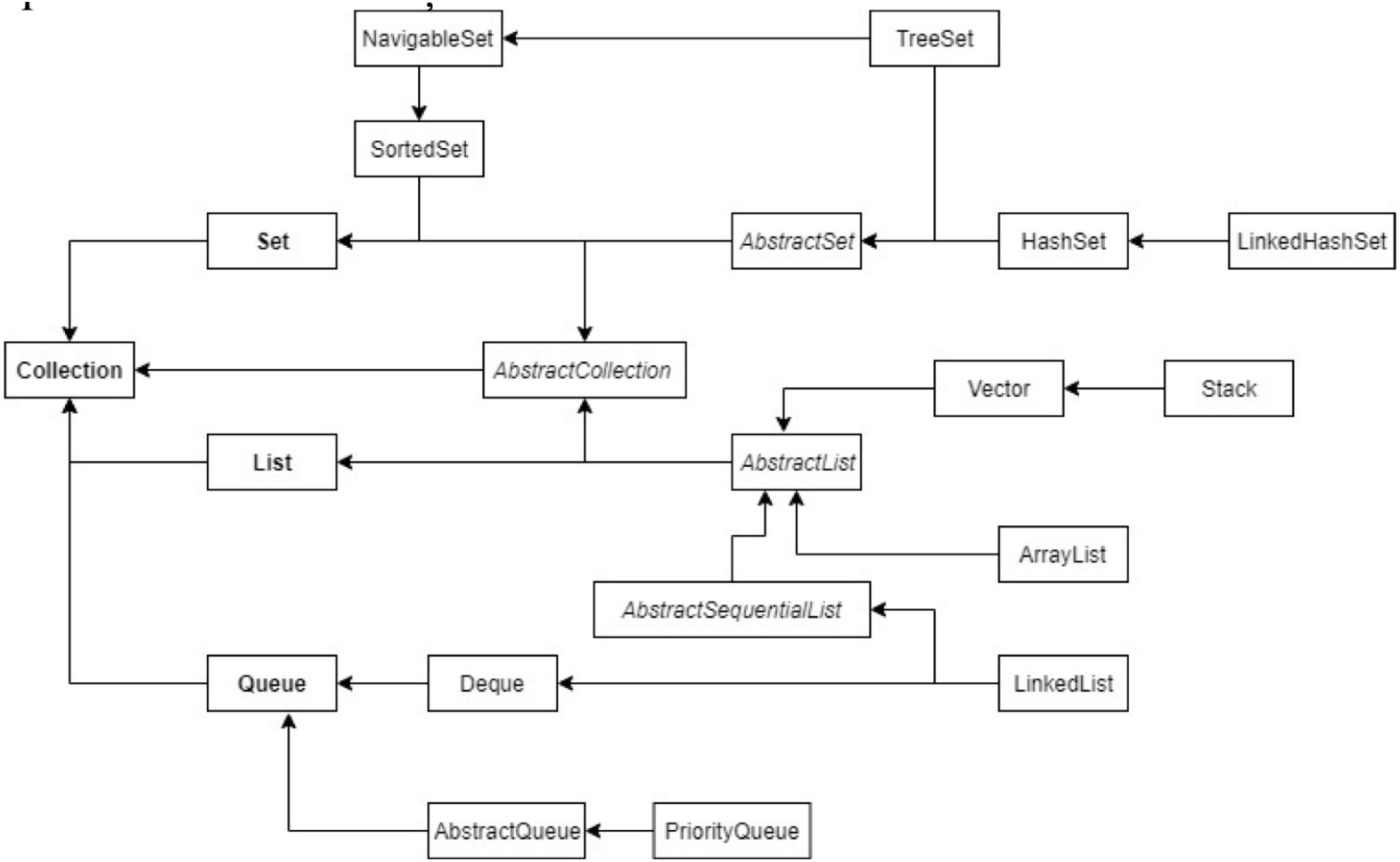
Wildcards Example

```
class Automobile {
    protected String type = "Automobile";
    public String getType() {
        return type;
    }
}

class Dacia extends Automobile {
    public Dacia() {
        type = "Dacia";
    }
}

class BMW extends Automobile {
    public BMW() {
        type = "BMW";
    }
}
```

/ Collections

Collections

- Java Collections Framework: collection, map
- 3 types of collections: Set, List, Queue
- Set – https://docs.oracle.com/javase/7/docs/api/java/util/Set.html
- List – https://docs.oracle.com/javase/7/docs/api/java/util/List.html
- Queue – https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html


NavigableSet
TreeSet
SortedSet
Set
AbstractSet
HashSet
LinkedHashSet
Collection
AbstractCollection
Vector
Stack
List
AbstractList
ArrayList
AbstractSequentialList
Queue
Deque
LinkedList
AbstractQueue
PriorityQueue

Collection Interface

- https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html
- Examples of defined methods:

```
boolean add(E e)

boolean addAll(Collection<? extends E> c)

boolean contains(Object o)

boolean remove(Object o)
```

Set Interface

- Set extends Collection. Elements must be unique

```
public class TestHashSet {
    public static void main(String[] args) {
        Set<String> set = new HashSet<>();
        set.add("Ana");
        set.add("Maria");
        set.add("Andrei");
        set.add("Mihai");
        set.add("Ana");

        System.out.println(set);
        Iterator<String> iterator = set.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
    }
}
```

Other Set Implementations

- *LinkedHashSet* – allows keeping the order used when inserting elements

- *TreeSet* – allows specifying an order to be used (ascending or descending)
- Uses the Comparable interface to compare elements with compareTo
- Implements a subinterface of Set called SortedSet

List Interface

- List extends Collection. Ordered collection that allows duplicates
- https://docs.oracle.com/javase/8/docs/api/java/util/List.html

- ArrayList stores elements in a dynamic array

- LinkedList stores elements in a linked list

/ Short Recap

Short Recap

- Primitive types, arrays
- Classes, objects + access modifiers
- Inheritance, overloading, casting
- Abstract classes, interfaces
- Internal classes
- Generics
- Collections

/ Practice in class

DogShelter

- Brigitte just received a donation of 2M USD to build a dog shelter in Romania.
- Follow these steps to write a program that creates and runs a dog shelter:

1. Create a class Shelter with the following properties: name (String), account (Account), animals (List of Animal)
2. Define the Account class with the following properties: id (int), iban (random String of 32 chars), currency (Enum), amount (double), created_date (date), interest (double)
3. Define an interface IShelter and declare the following methods: getName (String), getLatitude (float), getLongitude (float), getOwner (String), receiveDonation (double), spend (double, String)

DogShelter

4. Create a class DogShelter which extends Shelter and implements IShelter

5. Create Animal class and make it generic. Add the following fields: name (String), age (double), foodHistory (double[]), isHungry (Boolean), isThirsty (Boolean)

6. Keep animals both in a linked list (to keep their added order) and in a TreeSet (to allow keeping them in a desired order (by age).

7. Allow the shelter to accept elements of type Dog or more specific using generics

/ Practice, practice, practice

Students Map with TreeSets

Define a *Student* class containing a `name` and `grade` property. Requirements:

- Create a constructor that receives the name and the grade of a student
- Implement method `public double getGrade()`
- Define a Map that hold keys from 0 to 10, corresponding to a rounded grade
- Each value stored in a key of a map will hold a TreeSet of Students, allowing storing them in descending order by grade. E.g.: for key 8 in the Map, values of the corresponding TreeSet could be `[Ionel: 8.49, Vasile: 7.50]`
- Test the class in a main method

MyList of Generics

- Define a class named `MyList`, which holds a list of elements using an array of elements of type `T`
- Define the following methods:

```
public MyList(dimension)
public void add(T element)
public void print()
public boolean lookup(T element)
```

- Test the class in a main method

/ Q&A


OBILE/ACADEMY