Session Goals

- User should understand the need of collections in Java.
- User should understand Integer class in Java and should be able to differentiate between int and Integer class.
- User should understand ArrayList Class In Java.
- User should understand the difference between arrays and ArrayList in Java.
- User should be able to declare, initialize and perform CRUD Operations on ArrayList in Java.

Java-111- Collections in Java

Session 9

Session Agenda

- Wrapper Classes
  - Integer
- Collections in Java
- package and import
- ArrayList (Dynamic array)
  - Methods

Wrapper classes - Primitive vs Class Type

- int vs Integer
  - Integer is a wrapper **class** for int
  - Methods like equals(), parseInt(), toString(), reverse(), rotateLeft(), rotateRight() ...
  - Class methods available for manipulation
- Similar wrapper classes exist for other primitives (Boolean, Character, Float, Long etc.)
- char array vs String (we have already seen String class)
  - String are class objects
  - Methods like contains(), equals(), length(), trim(), indexOf(), charAt(), replace(), substring() ….
  - Add two Strings using +

Curious Cats 

- Does Integer take the same amount of space as int

Documentation - https://docs.oracle.com/javase/9/docs/api/java/lang/Integer.html

Activity- Integer (Use file from repl)

```
class IntegerTest {
        public static void main(String args[])  {
          Integer a = 123;
         Integer b = Integer.valueOf(456);
          int c = Integer.parseInt("789");
          int d = Integer.MAX_VALUE;
          Integer e = new Integer(20);
          String eStr = e.toString();
          System.out.println (a);
          System.out.println (b);
          System.out.println (c);
          System.out.println (d);
          System.out.println (eStr);
        }
}
```

Activity: Convert into an integer

Link

What are “Java Collections”?

- Collections are **libraries** for various **Data Structures** in Java, which need to represent **a group of elements**.
- Java provides these **utilities** for programmers in the form of libraries (just ***import*** and start using them. **ArrayList was one of them.**
- Using these predefined libraries and methods **reduces development effort**.
- Being familiar with the **common Collections** and their **methods** is important for a Java Developer.
- Knowing **when to use which** Collection is important. Also very useful for DSA.

Further Reading - https://www.journaldev.com/1260/collections-in-java-tutorial

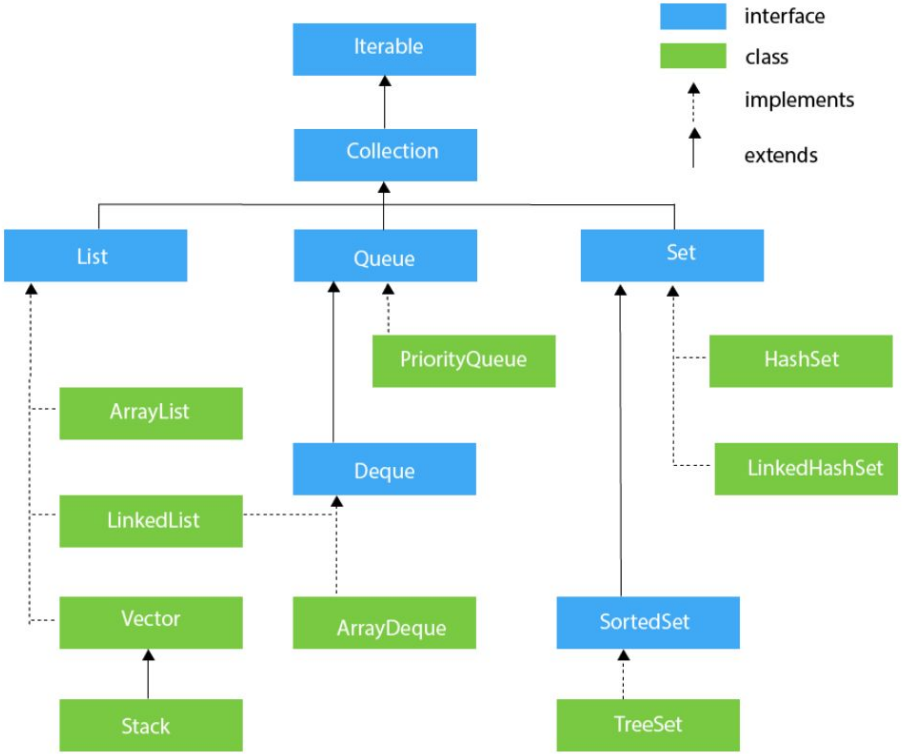
What are “Java Collections”?

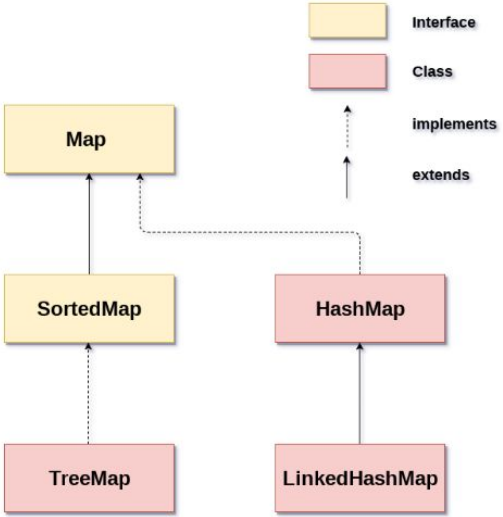
- Collection - **group of elements**
  - Basic Operations needed for any group of elements - **Insert, Update, Delete, Search**
  - There might be **different types of Collections**/Data Structures which need **specific properties** imposed on the group of elements
    - Ordered elements (**List**, **Queue**). *(We have already visited **ArrayList** in previous sessions)*
    - No duplicates (**Set**)
    - Elements have a mapping to another value (**Map**)

We will **focus on ArrayList, HashMap and HashSet** since these are the most frequently used ones.

Java Collections Framework

Key takeaway: Inheritance based hierarchy for Collections. Some common methods across Collections & some specific methods





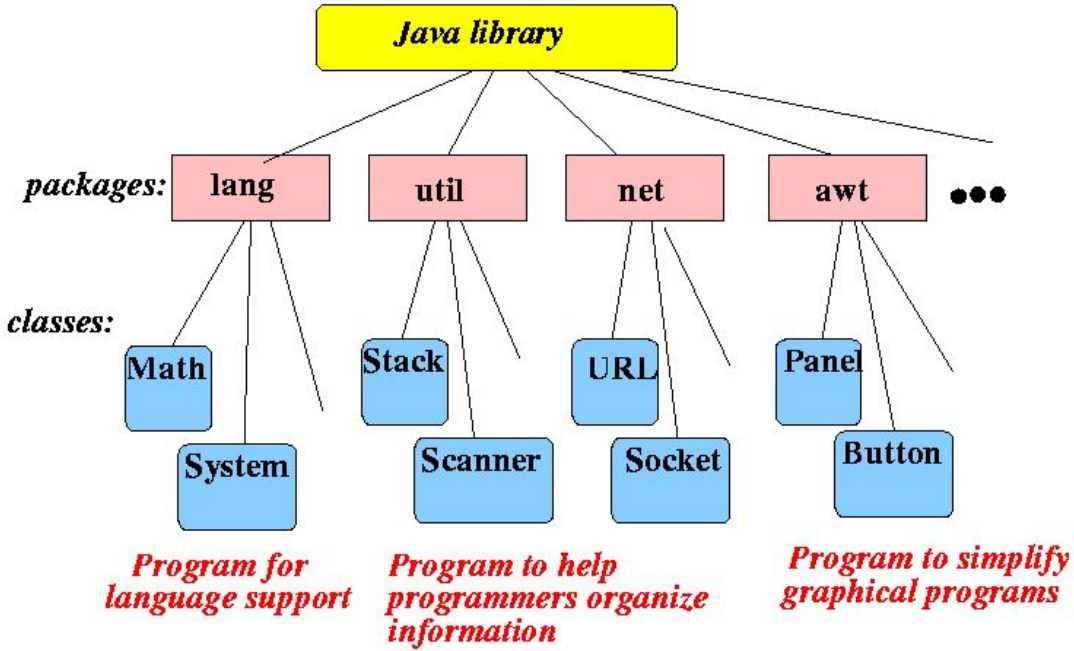
Source: javapoint.com

Java packages

- What is a library / **package** in Java?
  - A package is a way to organize related functionality or code in a single place in Java.
  - Consists of a **set of classes** that can be imported and used in other pieces of code.
- How to use a package?
  - Use **import** to include the package in your code
  - Then invoke methods on those package classes
- Standard Java Packages/Libraries
  - Java.lang (.math, .System etc.) - Remember the System.out you've been using all along?
  - Java.util (.Random, .Scanner etc.)

We'll revisit Packages in more detail in the next sprints.

Some Java Packages



ArrayList in Java (Dynamic array)

- **java.util.ArrayList** is a **Collection** (Data Structure **Class**) which is widely used for **Array Operations.**
  - Provides inbuilt methods for us to use, making life easy. This is missing in Array which operates using [index].
- ArrayList Object is initialized by the size. However, the **size grows automatically** if the collection **grows, or shrinks** if the objects are removed from the collection. Array is of fixed size.
- ArrayList Object allows us to **randomly access** the list.
- ArrayList **cannot be used for primitive types, like int, char**, etc. We need a wrapper class for such cases. Array can support either primitive types or objects.

We'll cover Collections in detail next week.

5 minute break

Create an ArrayList

Ways to create an ArrayList:

1. **ArrayList():** This constructor is used to build an empty array list.

   ```
   ArrayList arr = new ArrayList();
   ```

2. Preferred way to create an ArrayList (specifying the type of data it will store):

   ```
   ArrayList<Integer> arrList = new ArrayList<Integer>();  // Creating Integer ArrayList using generics
   ```

ArrayList Methods (Use file from repl)

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

Let’s use documentation to understand the methods in the ArrayList package.

CRUD operations are common for any data - Create, Read, Update and Delete.

Think of **Real World examples** where you would need an ArrayList and CRUD operations for it.

Recap - 6 Step Strategy

1. ***Understand the problem*** *(ask questions and get clarity)*
2. ***Design test data/test cases*** *(input and expected output)*
3. ***Derive the solution - solve the problem*** *(write pseudo code)*
4. ***Test the solution*** *(against the test data/case - dry run)*
5. ***Write the program/code*** *(using Java here)*
6. ***Test the code*** *(syntax errors, run time errors, logical errors)*

Activity: Change password

Link

What will be your approach to the problem? (Step 3)

Quickly put your answers in the chat!

Sort an ArrayList

- Use **Collections.sort()** method
- The **java.util.Collections** library has static methods that are applicable to all Collections including ArrayList.

```
import java.util.ArrayList;
import java.util.Collections;

public class JavaExample {
  public static void main(String[] args) {
    ArrayList<String> fruits = new ArrayList<String>();
    fruits.add("Orange");
    fruits.add("Apple");
    fruits.add("Banana");
    fruits.add("Pineapple");
    Collections.sort(fruits);

    for (String str : fruits) {
      System.out.println(str);
    }
  }
}
```

Search an ArrayList

- Use the **ArrayList.contains(element)** method

Further Reading

- Wrapper Classes
- Integer
- ArrayList

Keep
Learning,
Keep
Coding.