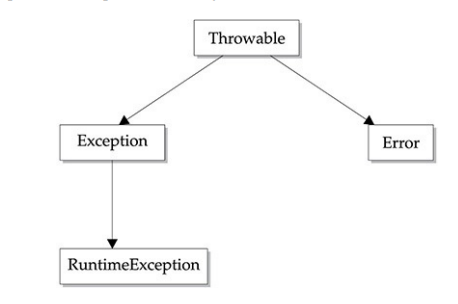
**OOPS**

Abstraction

1. Encapsulation
2. Inheritance
3. Polymorphism (overloading)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Same C | Private | Default | Protected | Public |
| Same P SC | Y | Y | Y | Y |
| Same P NonSC |  | Y | Y | Y |
| Diff P SC |  |  | Y | Y |
| Diff P NonSC |  |  |  | Y |

**Exceptions**



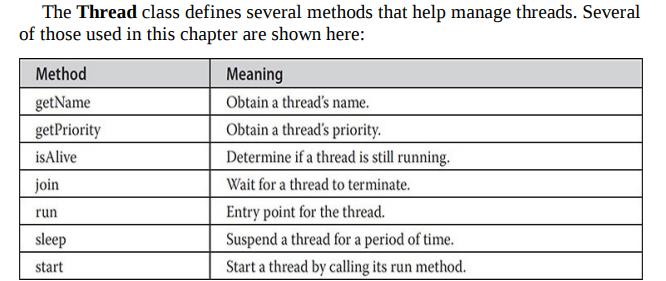
**Runtime Exception** (Unchecked)

Arithmetic / ArrayIndexOutfbounds / Classcast / IllegalArgument/ NumberFormat

Checked Exception

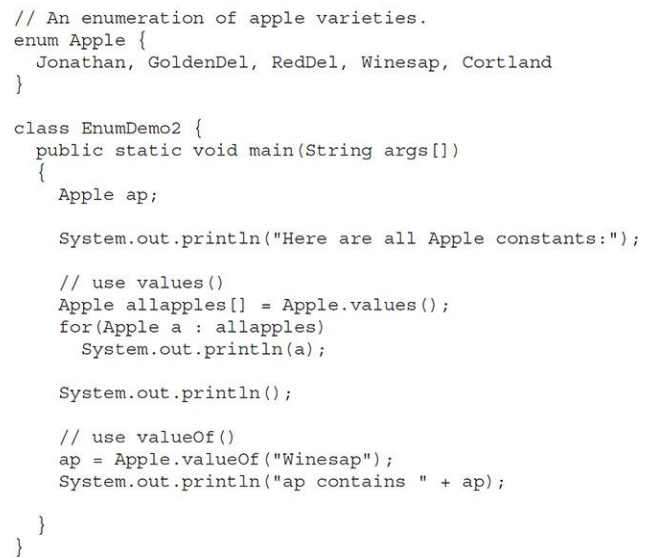
ClassNotFound / IllegalAccess / NoSuchField / NoSuchMethod

**Thread**

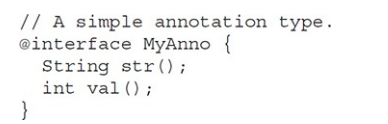


* Implements Runnable
* Externds Thread

**Enumeration**



**Annotations** (Embed supplemental info to source file)



1. Marker annotations
2. Single value annotations
3. Full annotations
4. Type annotations
5. @Target

@**Target(ElementType**.TYPE)

TYPE, FIELD, METHOD, CONSTRUCTOR, LOCAL\_VARIABLE, ANNOTATION\_TYPE, PARAMETER

@

interface MyAnnotation{

int value1();

String value2();

}

@Target({ElementType.TYPE, ElementType.FIELD, ElementType.METHOD})

@interface MyAnnotation{

int value1();

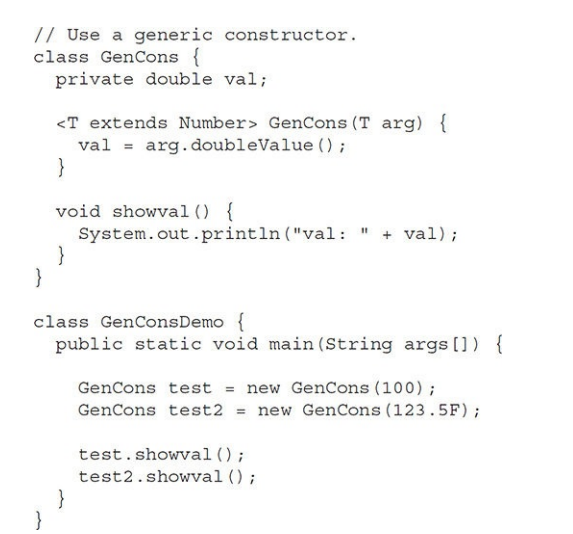
}

**@Retention** annotation is used to specify to what level annotation will be available.

**RetentionPolicy**.SOURCE/CLASS/RUNTIME

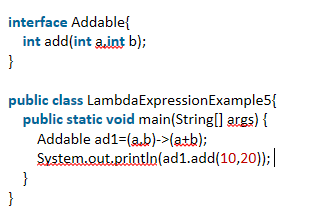
**Generics**

Paramterized types



**Lambda Expressions**

To represent one method interface using an expression.



**String**

|  |  |
| --- | --- |
| **Character Extraction** | **StringBuffer** |
| char charAt(int where) | StringBuffer( ) |
| getChars( ) | StringBuffer(int size) |
| getBytes( ) | StringBuffer(String str) |
| toCharArray( ) | StringBuffer(CharSequence chars) |
| **String Comparison** | int length( ) |
| equals( ) and equalsIgnoreCase | int capacity( ) |
| regionMatches( ) |  |
| startsWith( ) and endsWith( ) | ensureCapacity( ) |
| compareTo( ) | setLength( ) |
| indexOf( ) - lastIndexOf( ) | charAt( ) and setCharAt( ) |
| substring( ) | getChars( ) |
| concat() | append( ) |
| replace() | insert() |
| trim() and strip() | reverse() |
| toUpperCase()  toLowerCase() | delete( ) and deleteCharAt( ) |
|  | replace( ) |
|  | substring( ) |

**Collections**

|  |  |
| --- | --- |
| iterator() | add(Object) |
| max() | addAll(Collection c) |
| parallelStream() | clear() |
| remove(Object o) | contains(Object o) |
| removeAll(Collection c) | containsAll(Collection c) |
| removeIf(Predicate filter) | equals(Object o) |
| retainAll(Collection c) | hashCode() |
| size() | isEmpty() |
| stream() | toArray() |

**Linkedlist**

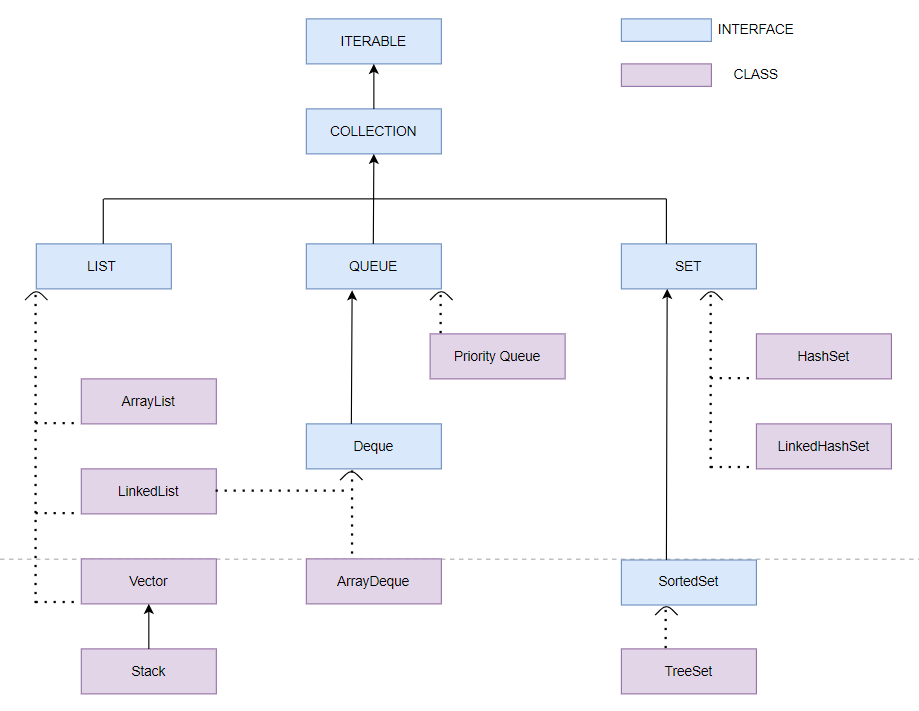
|  |  |
| --- | --- |
| addFirst() | removeLast() |
| addLast() | getFirst() |
| removeFirst() | getLast() |

**Collections**

**-** Utility class

- Collections.addAll(arrlist, "web", "site");

- Collections.sort(arrlist);



**Streams**

Input -> Operations -> Terminal operation -> Output

String[] arr = { "Geeks", "for", "Geeks" };

Stream<String> stream = **Arrays.stream**(arr);

stream.forEach(str -> System.out.print(str + " "));

Stream<string> stream = **Stream.of**("Geeks", "for", "Geeks");

Integer[] empIds = { 1, 2, 3 };

List<Employee> employees = Stream.of(empIds)

.**map**(employeeRepository::findById)

.**collect**(Collectors.toList());

.**filter**(e -> e.getSalary() > 200000)

#### findFirst, toArray, flatMap Lazy Evaluation – On terminal operation sorted / min and max / distinct .reduce(0.0, Double::sum);

#### java.util.stream.Stream Stream.concat(stream1,stream2).foreach()

Integer[] numbers = new Integer[] { 1, 2, 3 };

List<Integer> list = **Arrays.asList**(numbers);

|  |  |
| --- | --- |
| **Java 8** | **Java 11** |
| Functional Interfaces | Http Client |
| Default methods | Local-Variable Syntax for Lambda |
| Optional | toArray |
| Datetime api | var |
| Stream | Removed java.transaction (JTA) |