Module#3 - Interfaces, Casting, Contracts

→ Interfaces - List of methods in a dedicated source file.
→ Classes implement interface

→ Interface name must match source filename
→ Semicolon instead of method body.
* cast wider var into shorter var type.

$\boxed{\text{byte} < \text{short} < \text{int} < \text{long} < \text{float} < \text{double}}$

small ← B S I L F D → big

Shallow equality:-
Compares two vars on the stack, not heap.

```
    public boolean equals(Star that) {
        Star that = (Star)x          // need casting.
      if (this.mass != that.mass)
           return false.
      if (this.age != that.age)
           return false
      return true
    }.
```

Hashing:-   hashCode()
- A fast efficient strategy for locating objects in memory.

contract= given any two objects 'x' & 'y' if x.equals(y) is true then x.hashCode() must equal y.hashCode()

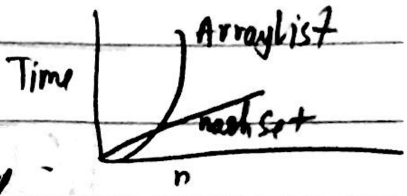
HashSet: java.util.HashSet.

```
    HashSet<Book> books = new HashSet<book>();
```

→ methods: add() & contains()
→ enhanced for loop. - arbitrary traversing

-> .add() adds specified element to this set if it is not already present.
-> HashSet need to implement hashCode() & equals().

Arraylist vs. HashSet:-



-> HashSet automatically checks for prior presence. & need to add hashCode & respect the contract.
-> HashSet is the most efficient, easy to use data structure.

-> A class is ordered if it implements Comparable<X>

```
public int compareTo(Xyz that) {...}
    <0  => this is less than that       // "this" comes before "that
    >0  => this is greater that that    // "this" comes after "that"
    0   => this.equals(that)            // "this" equals that
```

best way to implement equals.

```
public boolean equals(Object x) {
    Whatever that = (Whatever) x
    return this.compareTo(that) == 0
}
```

<u>TreeSets</u>:

-> traversal order is determined by .compareTo()

-> Pass [ArrayList, HashSet, TreeSet] as an arg. to ctor of [ArrayList, HashSet, Tree]