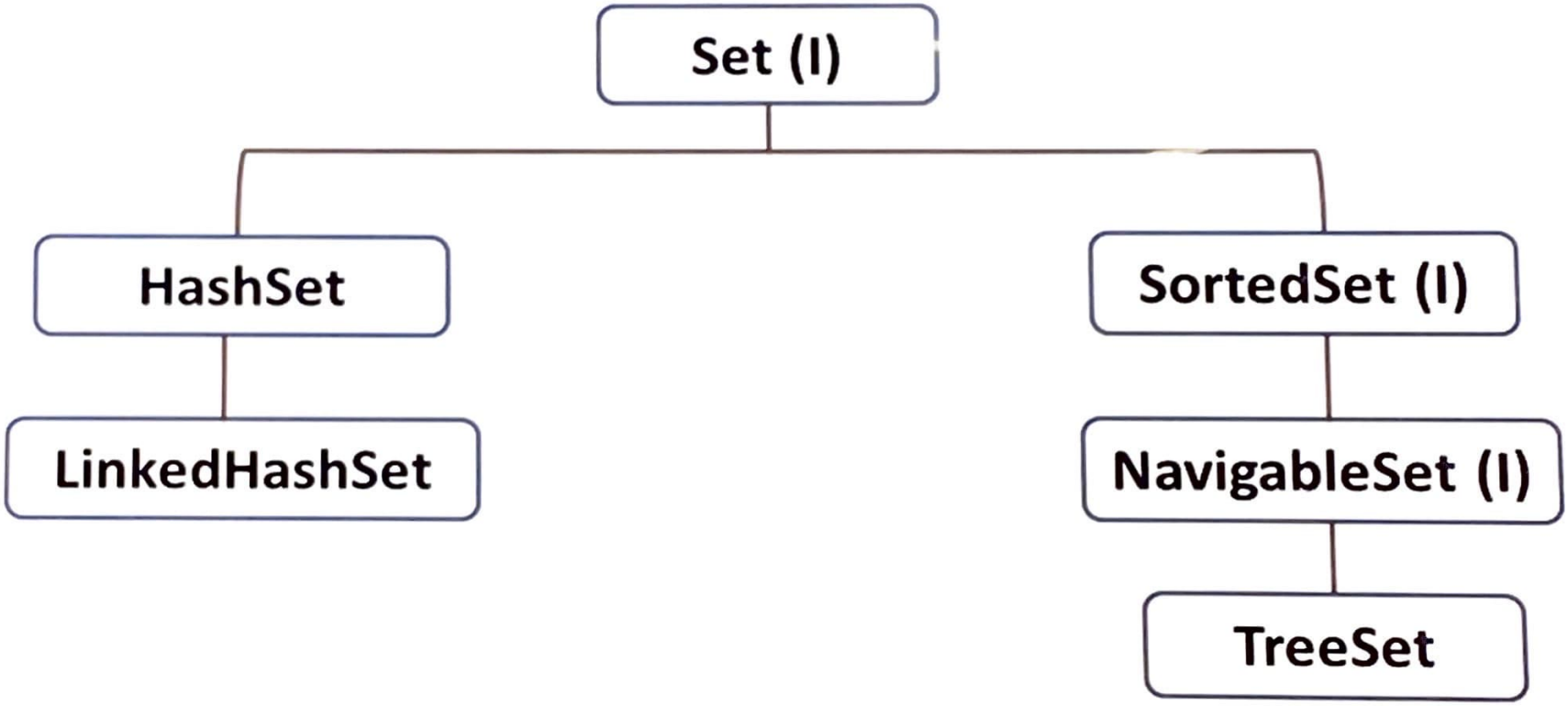
Set Interface

Set Interface Implemented Classes



Set(I)

- Set is a child interface of **Collection**.
- If we want to represent a group of individual objects as a single entity where **duplicates** are not allowed & the insertion order is **not preserved**.
- Set interface doesn't contain any new method and we have to use only **Collection** interface method.

HashSet - Constructors

public HashSet(): Constructs a new empty set with default initial capacity 16 and fill ratio (load factor) is 0.75

public HashSet(Collection c): The new HashMap is created with a default load factor (0.75) and an initial capacity

public HashSet(int initialCapacity): Constructs a new empty set with specified initial capacity

public HashSet(int initialCapacity, float loadFactor): Constructs a new, empty set with specified initial capacity and the specified load factor

HashSet

- The underlying data structure is **Hashtable**.
- **Duplicate** objects are not allowed.
- Insertion order is **not preserved** and it is based on **Hashcode** of the objects.
- **null** insertion is possible only once.
- **Heterogeneous** objects are allowed.
- Implements **Serializable, Cloneable** but not **RandomAccess** interface.
- **HashSet** is the best choice if our frequent operation is **search** operation.

HashSet

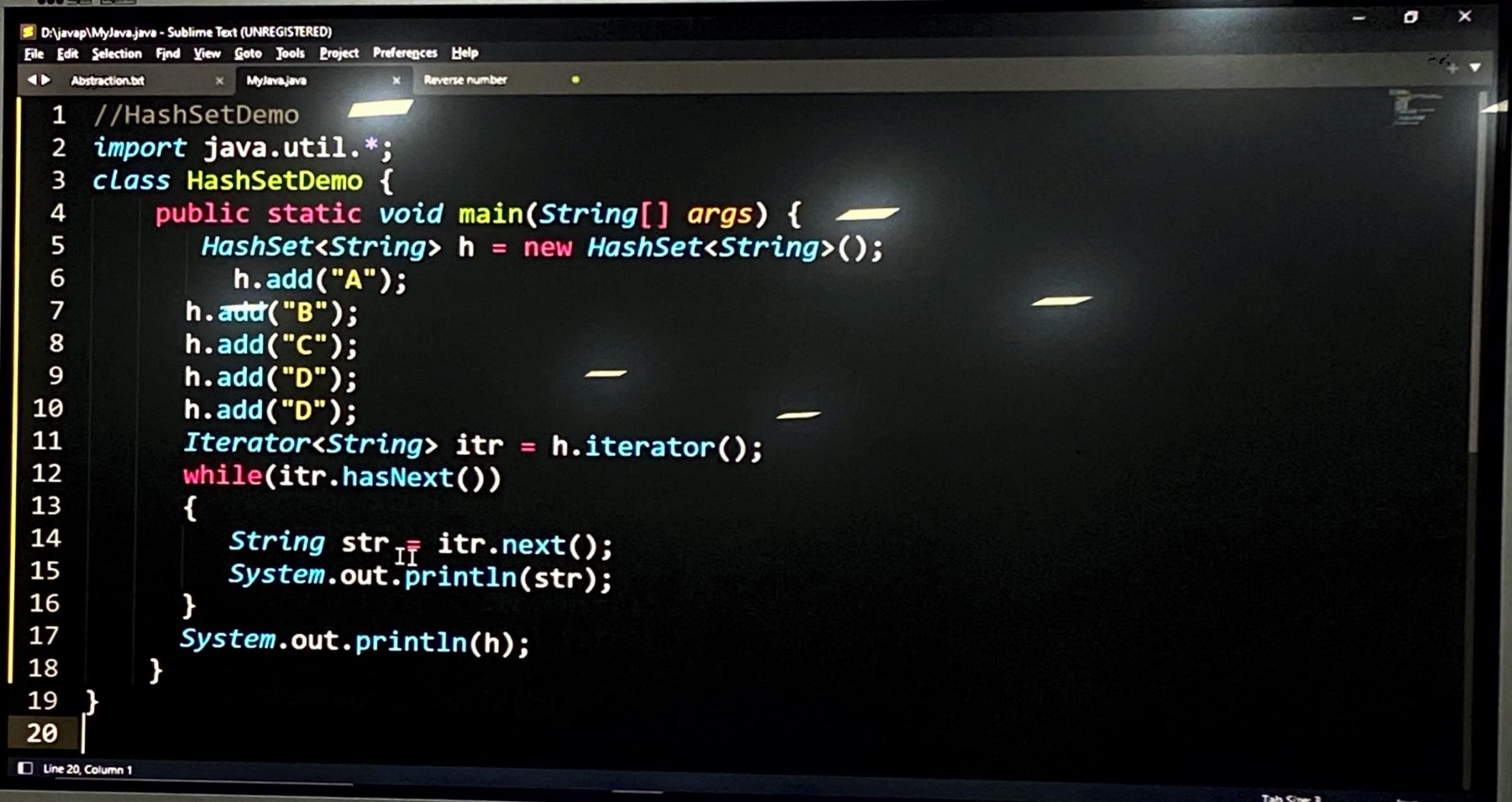
- If we are trying to insert a duplicate object then we won't get any **Exception** & the **add**() method simply returns "false".

```
HashSet h=new HashSet();
        h.add("A");          true
        h.add("A");          false //duplicate
```

Fill ratio or load factor:

After filling how much ratio a new **HashSet** object will be created, this ratio is called the **fill ratio** or **load factor**.

Ex: fill ratio 0.75 means after filling 75% a new **HashSet** object will be created.


D:\javap\MyJava.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
Reverse number
1 //HashSetDemo
2 import java.util.*;
3 class HashSetDemo {
4 public static void main(String[] args) {
5 HashSet<String> h = new HashSet<String>();
6 h.add("A");
7 h.add("B");
8 h.add("C");
9 h.add("D");
10 h.add("D");
11 Iterator<String> itr = h.iterator();
12 while(itr.hasNext())
13 {
14 String str = itr.next();
15 System.out.println(str);
16 }
17 System.out.println(h);
18 }
19 }
20
Line 20, Column 1

SortedSet

SortedSet

- It is a child interface of a **Set** Interface.

- If we want to represent a group of individual objects according to some **sorting order** where dupiicate objects are not allowed then we should go for **SortedSet**.

SortedSet Methods

public Comparator comparator(): Use for the natural ordering of its elements.

public Object first(): Returns the first (lowest) element

public Object last(): Returns the last (highest) element

public SortedSet headSet(Object toElement): Returns a element whose elements are less than **toElement**

public SortedSet tailSet(Object fromElement): Returns a element whose elements are greater than or equal to **fromElement**

public SortedSet subSet(Object fromElement, Object toElement): Returns elements whose elements range >= **fromElement** but < **toElement**

TreeSet

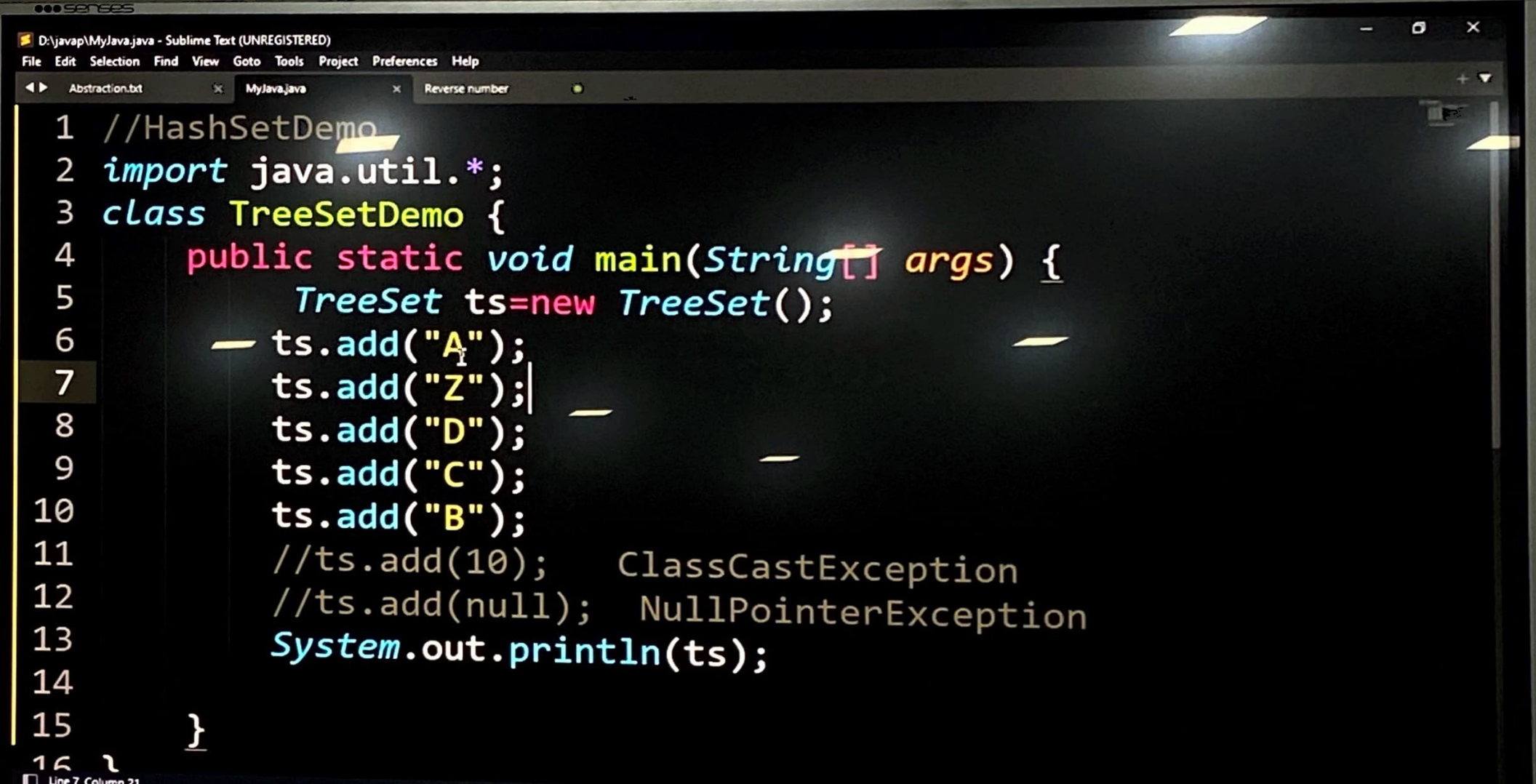
TreeSet

- Underlying Data Structure is a **balanced tree**.
- Duplicates are not allowed.
- All objects will be inserted based on some sorting order, It may be **default natural sorting** order or **customize sorting** order.
- Heterogeneous objects are not allowed.
- **null** insertion is possible (only once).
- **TreeSet** implements **Serializable** & **Cloneable** but not **RandomAccess**.

TreeSet - Constructors

- **public TreeSet():** Constructs a new, empty TreeSet, sorted according to the natural ordering of its elements.
- **public TreeSet(Collection c):** Constructs a new TreeSet containing the elements in the specified collection
- **public TreeSet(Comparator comparator):** Constructs a new, empty TreeSet, sorted according to the specified comparator.
- **public TreeSet(SortedSet s):** Constructs a new TreeSet containing the same elements and using the same ordering as the specified sorted set.

Ex:


D:\javap\MyJava.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
Reverse number
1 //HashSetDemo
2 import java.util.*;
3 class TreeSetDemo {
4 public static void main(String[] args) {
5 TreeSet ts=new TreeSet();
6 ts.add("A");
7 ts.add("Z");
8 ts.add("D");
9 ts.add("C");
10 ts.add("B");
11 //ts.add(10); ClassCastException
12 //ts.add(null); NullPointerException
13 System.out.println(ts);
14
15 }
16 }
Line 7, Column 21