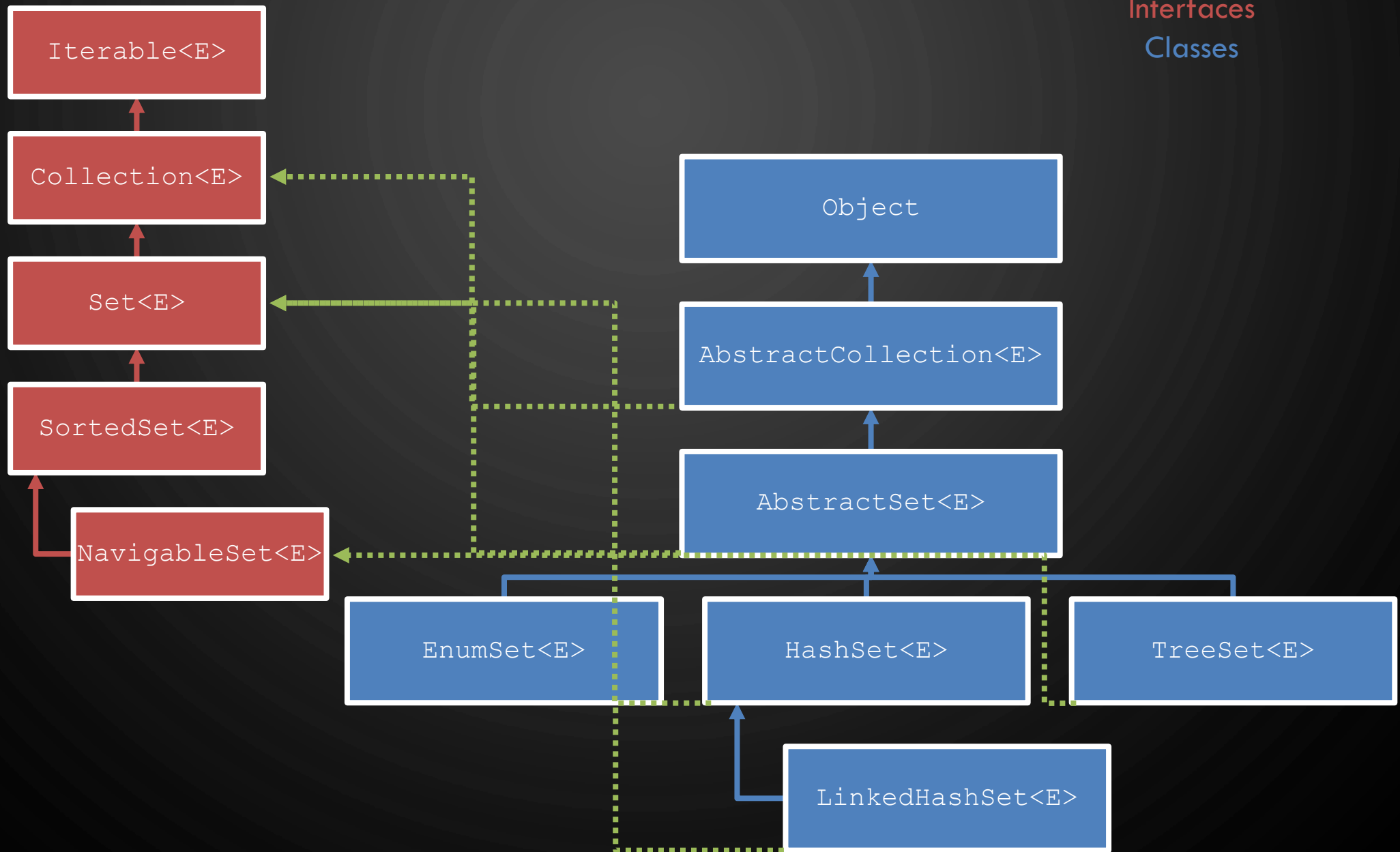




JAVA SETS AND MAPS

SUMMARY OF CLASSES (SETS)

- **EnumSet<E>** - Specialized set for enumerated types. Implemented as bit-vector.
- **HashSet<E>** - Set implemented with chained hash table – no guaranteed iteration order
 - **LinkedHashSet<E>** - Same, but with guaranteed iteration order. No complexity overhead
 - ***E* should override both `hashCode()` and `equals()`! Default `hashCode()` hashes memory address of object (typically, not always) and default `equals()` compares memory address**
- **TreeSet<E>** - Set implemented with red-black tree. Needs `Comparator<E>` or `E` should implement `Comparable<E>`
- There are no multisets in the Java library. Some external libraries have them.
- To find how to use them, go to the Java API!



EXAMPLE OF USING SET<E>

```
1. Scanner s = new Scanner(new File("numbers.txt"));
2. HashSet<Integer> numbers = new HashSet<>();
3. while (s.hasNextInt())
4.     numbers.add(s.nextInt());
5. ...elsewhere...
6. int sumOfUnique = 0;
7. for (Integer i : numbers)
8.     sum += i;
```

EXAMPLE OF OVERRIDING HASCODE()

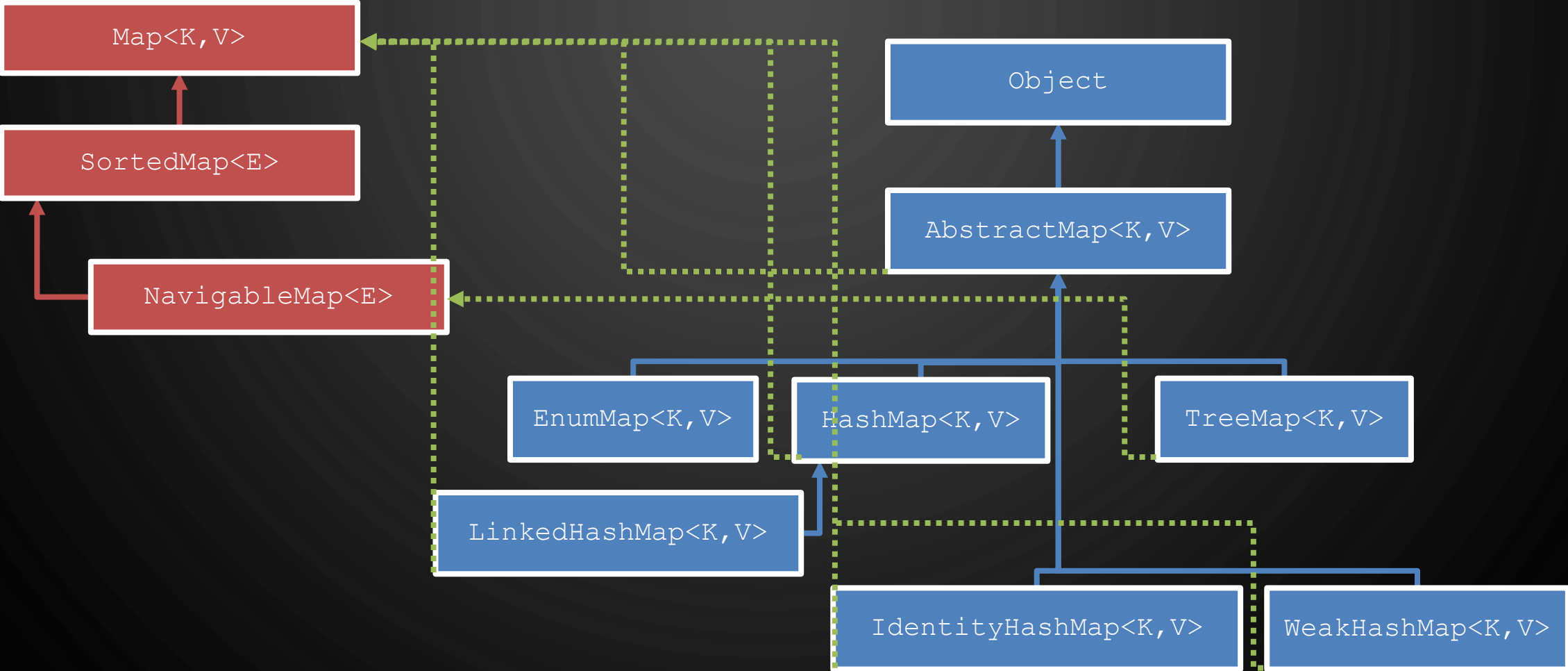
```
1.  public class Student {
2.      String id; //unique!
3.      String name;
4.      List<Courses> courses;
5.      ...stuff...
6.      public boolean equals(Object obj) {
7.          if(obj != null && obj instanceof Student)
8.              return id.equals(((Student)obj).id);
9.          return false;
10.     }
11.     public int hashCode() {
12.         return id.hashCode();
13.     }
14. }
```

[Really great stackoverflow answer about creating hash codes.](#)

SUMMARY OF CLASSES (MAPS)

- **EnumMap<K, V>** - Specialized set for enumerated types. Implemented as bit-vector.
- **HashMap<K, V>** - Set implemented with chained hash table – no guaranteed iteration order
 - **LinkedHashMap<K, V>** - Same, but with guaranteed iteration order. No complexity overhead
 - *K should override both hashCode () and equals ()! Default hashCode () hashes memory address of object (typically, not always) and default equals () compares memory address*
- **TreeMap<K, V>** - Set implemented with red-black tree. Needs Comparator<K> or K should implement Comparable<K>
- **IdentityHashMap<K, V>** - hash map specifically for objects that use reference equality instead of object equality
- **WeakHashMap<K, V>** - Like a hash map, but does not prevent garbage collector from removing keys
- There are no multisets in the Java library. Can fake with Map<K, List<V>>
- To find how to use them, go to the Java API!

Interfaces
Classes



EXAMPLE OF USING MAP<K,V>

```
1. Scanner s = new Scanner(new File("numbers.txt"));
```

```
2. HashMap<Integer,String> numbers = new HashMap<>();
```

```
3. while (s.hasNextInt())
```

```
4.     numbers.put(s.nextInt(), s.next());
```

```
5. ...elsewhere...
```

```
6. System.out.println(numbers.get(5)); //get a value
```


PROBLEM

- Design a barcode scanner for a super market: Given the barcodes of the items in your shopping list, look up in the inventory database for its availability and if available print the quantity.
- What will be the underlying container?
- Read the inventory data (item barcode, quantity, per item cost) from the file given in the website.
- Bonus: Prepare the receipt (total cost) for the shopping list.
 - You may format a shopping list any way you want. Be sure to handle the case where the item does not exist!