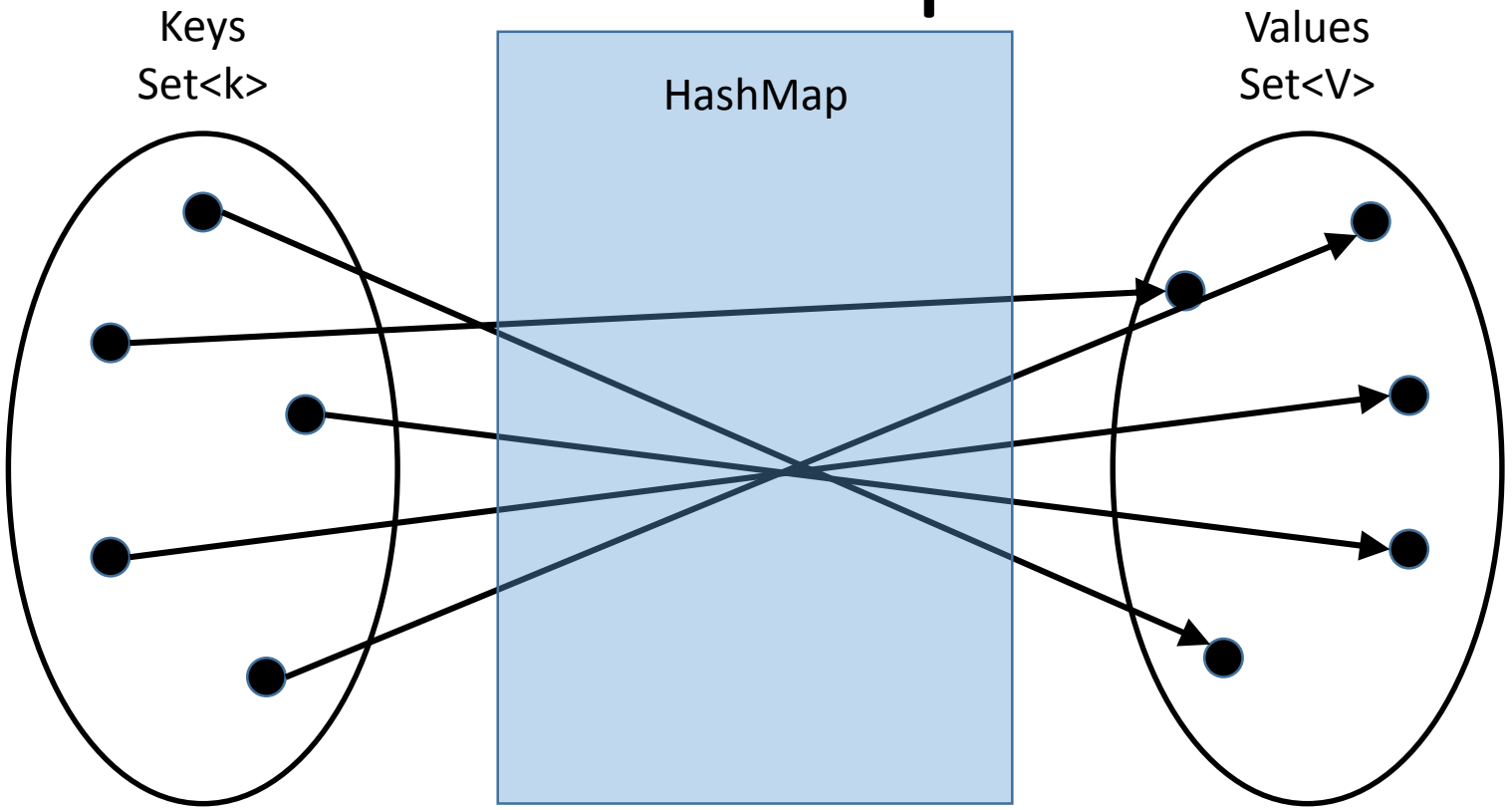
CS 2334: Lab 8
HashMaps and Enums

HashMap

- HashMap<K, V> where K is a key and V is the type of stored values.
- Maps a given key to a value.
- Very efficient means of storage and lookup.
- Objects are added with put(key K, value V).
- Objects are looked up with get(key K).

Note: keys and values are generics, so they can be any type of Object.

HashMap



A HashMap is the function between two unordered Sets. It tells the computer what the relation between a given key and value is.

HashMap

```
import java.util.HashMap;

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

numbers.put("two", 2);
numbers.put("one", 1);
numbers.put("twenty", 20);

System.out.println(numbers.get("two"));  //This will print 2
```

HashMap

In order to iterate through a HashMap, you have to use a foreach loop. There are two ways to do this:

1. Loop over the keys
2. Loop over the values themselves

HashMap

```
import java.util.HashMap;

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

numbers.put("two", 2);
numbers.put("one", 1);
numbers.put("twenty", 20);

for (String key : numbers.keyset()){
      System.out.println(key);                    //“two”
}                                                 //”one”
                                                  //”twenty”
```

HashMap

```
import java.util.HashMap;

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

numbers.put("two", 2);
numbers.put("one", 1);
numbers.put("twenty", 20);

for (Integer value: numbers.values()){
      System.out.println(value);                    //   2
}                                                   //  1
                                                    //  20
```

HashMap

For more information about HashMaps, check out the API: http://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

Sets vs Lists

```
for (String key : numbers.keyset()){                    //“two”
      System.out.println(key);                          //”one”
}                                                       //”twenty”
```

- Sets are unordered and cannot have duplicates
- What happens when we try numbers.put(“two”, 22); ?

Sets vs Lists

```
for (String key : numbers.keyset()){                    //“two”
      System.out.println(key);                          //”one”
}                                                       //”twenty”
```

- Sets are unordered and cannot have duplicates
- What happens when we try numbers.put(“two”, 22); ?
  - The original “two” entry is replaced with this new one
  - Equality is defined by the key class equals() method

Enumerated Data Types

- Recall from last week's lab that enumerated data types allows a variable to be one of a set of predefined constants.
- Example: Planets

```
public enum Planet{
        MERCURY, VENUS, EARTH, MARS, JUPITER, SATURN, URANUS,
        NEPTUNE, PLUTO;
}
```

Enumerated Data Types

- We can also give more information to these enums.

```
public enum Planet{
        MERCURY(1), VENUS(2), EARTH(3), MARS(4), JUPITER(5),
        SATURN(6), URANUS(7), NEPTUNE(8), PLUTO(9);

        private int position;
        private Planet(int position){
                this.position = position;
        }
}
```

Lab 8: States

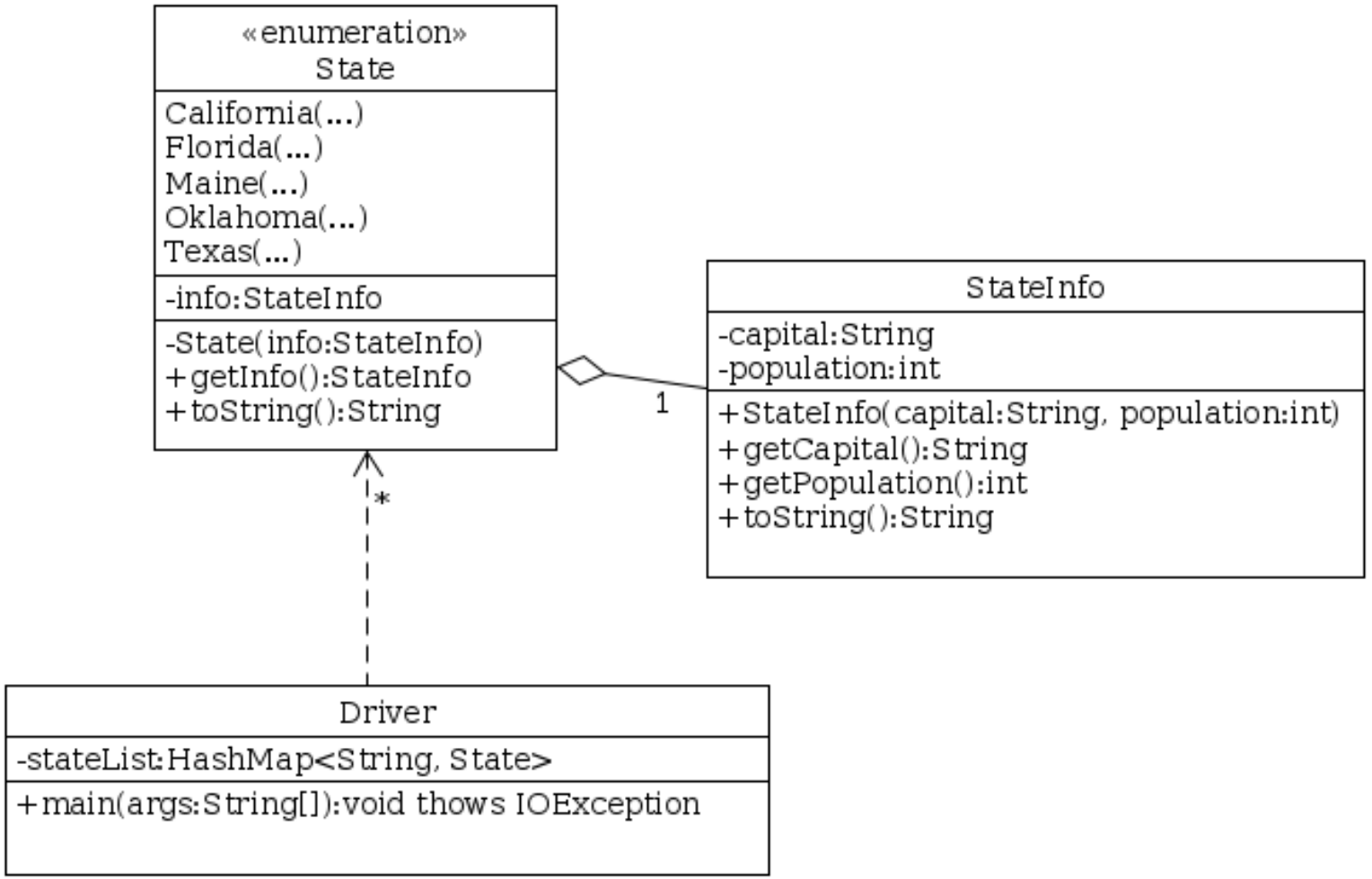
- This lab will ask for a state choice from the user and then print out information regarding the chosen state.
- The information is stored in a HashMap using the state's abbreviations as keys and an enum as the values.
- The enum stores the capital and population of the state in a StateInfo object.

Lab 8: States

Given a UML diagram that describes information about states:

- Implement each class, including the specified instance variables and methods
- Implement testing procedures for the classes

States


«enumeration»
State
California(...)
Florida(...)
Maine(...)
Oklahoma(...)
Texas(...)
-info:StateInfo
-State(info:StateInfo)
+getInfo():StateInfo
+toString():String
1
StateInfo
-capital:String
-population:int
+StateInfo(capital:String, population:int)
+getCapital():String
+getPopulation():int
+toString():String
*
Driver
-stateList:HashMap<String, State>
+main(args:String[]):void thows IOException

Lab 8 Notes

- Create each class in the UML diagram
  - Include all methods and instance variables, with the specified visibility
  - Watch spelling and casing
  - Use the default package
- Implement attributes and methods
  - Classes are dependent on each other, so you can have temporary errors while you implement

Submission

- Submit only one file: lab8.zip (casing matters)
- Due date: Friday, October 16th @11:59pm
- Submit to lab8 dropbox on D2L