7.3 HashMap

HashMap

Arrays and Lists store elements as ordered collections, with each element given an integer index.

HashMap is used for storing data collections as key and value pairs. One object is used as a key (index) to another object (the value).
The **put**, **remove**, and **get** methods are used to add, delete, and access values in the HashMap.

Example:

```
import java.util.HashMap;
public class MyClass {
 public static void main(String[ ] args) {
   HashMap<String, Integer> points = new HashMap<String, Integer>();
   points.put("Amy", 154);
   points.put("Dave", 42);
   points.put("Rob", 733);
   System.out.println(points.get("Dave"));
 }
}
// Outputs 42
```

We have created a HashMap with Strings as its keys and Integers as its values.

Note:

Use the **get** method and the corresponding key to access the HashMap elements.

Q: What is the output of this code?

```
import java.util.HashMap;
class A {
  public static void main(String[ ] args) {
  HashMap<String, String> m = new HashMap<String, String>();
  m.put("A", "First");
  m.put("B", "Second");
  m.put("C", "Third");
  System.out.println(m.get("B"));
  }
}
```

- First
- Third
- Second
- Nothing

HashMap

A **HashMap** cannot contain duplicate keys. Adding a new item with a key that already exists overwrites the old element.

The HashMap class provides **containsKey** and **containsValue** methods that determine the presence of a specified key or value.

If you try to get a value that is not present in your map, it returns the value of **null**.

Note:

null is a special type that represents the absence of a value.

Q: Fill in the blanks to declare a HashMap, add two items, and print one of them.

```
HashMap<String, String> m =
   _____ HashMap<String, String>();
m.put("A", "First");
m.____("B", "Second");
System.out.println(m._____("B"));
```