# CS139 HashMaps

## The Problem

- We want to store the account balances for everyone at JMU with an eid.
- We *have* the eid, we want to be able to *find* the account balance quickly.
- How can we accomplish this using the tools we've seen so far???

## One Solution: Parallel Arrays

```java
String[] eids;
double[] balances;

eids = new String[NUM_ACCOUNTS];
balances = new double[NUM_ACCOUNTS];

eids[0] = "bernstdh";
balances[0] = 3.25;

eids[1] = "bowersjc";
balances[1] = 0.0;

eids[2] = "spragunr";
balances[2] = 223.18;
```

# One Solution: Parallel Arrays

```
String[] eids;
double[] balances;

eids = new String[NUM_ACCOUNTS];
balances = new double[NUM_ACCOUNTS];

eids[0] = "bernstdh";
balances[0] = 3.25;

eids[1] = "bowersjc";
balances[1] = 0.0;

eids[2] = "spragunr";
balances[2] = 223.18;
```

- Given “spragunr” how can we retrieve 223.18?

  Write a method with this header:

```
public static double getBalance(String eid, String[] eids,
                                double[] balances)
```

## Parallel Array Lookup

```
public static double getBalance(String eid, String[] eids,
                                 double[] balances) {

    for (int i = 0; i < eids.length; i++) {

       if (eids[i].equals(eid)) {
          return balances[i];
       }

    }
    return -1;
 }
```

## Problems with Parallel Arrays

- Can you think of any problems with this approach?
  - Imagine that there are several million active eid's.

## Problems with Parallel Arrays

- Can you think of any problems with this approach?
  - Imagine that there are several million active eid's.

- Slooow. Lookup may require us to examine all entries.
- Awkward to add new entries (arrays have a fixed-size)

## Java Collections

- Java provides an assortment of collection classes:
  - https://docs.oracle.com/javase/tutorial/collections/
  - You are familiar with ArrayList

## Maps

- A map is a collection type that stores a mapping from keys to values.
  - In our example the eid is the key, the account balance is the value
- Also called:
  - Dictionary
  - Associative array

# HashMap Example

```java
import java.util.HashMap;

public class HashMapDemo {

   public static void main(String[] args) {

      HashMap<String, Double> balances;

      balances = new HashMap<String, Double>();

      balances.put("spragunr", 223.18);
      balances.put("bowersjc", 0.00);
      balances.put("bernstdh", 3.25);


      // Look up a balance:

      System.out.println("BALANCE IS: " + balances.get("spragunr") );

   }
}
```

## HashMap Example

Key type

Value type

```
import java.util.HashMap;

public class HashMapDemo {

   public static void main(String[] args) {

      HashMap<String, Double> balances;

      balances = new HashMap<String, Double>();

      balances.put("spragunr", 223.18);
      balances.put("bowersjc", 0.00);
      balances.put("bernstdh", 3.25);


      // Look up a balance:

      System.out.println("BALANCE IS: " + balances.get("spragunr") );

   }
}
```

## HashMap Example

```java
import java.util.HashMap;

public class HashMapDemo {

   public static void main(String[] args) {

      HashMap<String, Double> balances;

      balances = new HashMap<String, Double>();

      balances.put("spragunr", 223.18);
      balances.put("bowersjc", 0.00);
      balances.put("bernstdh", 3.25);


      // Look up a balance:

      System.out.println("BALANCE IS: " + balances.get("spragunr") );

   }
}
```

Key

Value

Key

## HashMap Efficiency

- Nice thing about HashMap:
  - Lookup time *doesn't* grow with the number of elements stored
  - Lookup is just as fast with a HashMap that has 1,000,000 keys as it is with 10

- Hashing – Something to look forward to in CS240!