Announcements

• Assignment 10
  • Due Thursday 12/01
  • tonight

• Assignment 11
  • Due Thursday 12/08
  • Optional/extra credit

• Code jam this week in lab
Key Concept review

primitive data type vs objects

8 primitive data types in java:
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

We’ve covered:
   ints, floats, doubles, booleans, chars
Didn’t cover:
   bits, shorts, longs
Primitive data types

8 primitive data types in java:
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

We’ve covered:
ints, floats, doubles, booleans, chars
Didn’t cover:
bits, shorts, longs
Strings

Strings are not primitives, they are ...
    they are objects!

What happens if we print an object?
    We see the location in memory?

Why does the following print:
    String name = “adam”;
    System.out.println(name);

Answer: The String class has a toString() method!
Recursion

What is the sum of the following numbers?

[10, 20, 30, 50, 300, 543, 553, 654, 7654, 7654, 34, 25, 673, 6753]

How would you solve this?
Recursion

What is the sum of the following numbers?
[10, 20, 30, 50, 300, 543, 553, 654, 7654, 7654, 34, 25, 673, 6753]

Approach 1 (iterative):
- keep track of a running sum
- add each number to the sum

How many computations/steps do you have to do?
atleast 14 – keep track, add every number
Recursion

What is the sum of the following numbers?

[10, 20, 30, 50, 300, 543, 553, 654, 7654, 7654, 34, 25, 673, 6753]

Approach 2 – lazy!:
- keep track of one number
- ask my friend to sum the rest of the numbers
- add the answer from my friend to the number I kept track of

How many computations/steps do you have to do?
1 or 2 – keep track, add once
ArrayList

• Convenient when we don’t know the size we need at the start

• Best for storing sequences/list of data

• When we run out of space, the array list resizes itself

• Adding elements to the end is generally fast (so long as we don’t need to resize)

• Removing elements or inserting in the middle can be slow (need to shift elements)
Hashmap

Stores <key, value> pairs
  Examples: associate a name to age
  Examples: associate a studentId to a grade

Fast lookup, add, and remove by key

Does not preserve the ordering of data

Keys should be unique
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    boolean test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    boolean test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    bool test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
Hashmap – iterating through hashmap

```java
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    boolean test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
```
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    boolean test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
Visualizing Hashmaps

```java
public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("dog", "woof");
    map.put("cow", "moo");
    map.put("cat", "meow");
    map.put("bird", "chirp");
    System.out.println(map.get("dog"));

    for (String key : map.keySet()) {
        System.out.printf("What does the %s say? %s\n", key, map.get(key));
    }

    boolean test = map.containsKey("turkey");
    System.out.println(test);

    map.remove("cat");
}
```
Exercise

Write a program, LetterCount.java, that counts the number of times each character appears in a given string.

```bash
$ java LetterCount
Please enter a word: lol
l: 2
o: 1

$ java LetterCount
Please enter a word: abba
a: 2
b: 2
```
Exercise

Write a program, Cake.java, that implements a Cake class that stores a cake name and cost. In main(), read in a CSV file of cakes into an ArrayList and sort them from least expensive to most expensive.

```
$ java-introcs Cake cakes.txt
Red velvet cake: $2.0
Chocolate cake: $3.5
Strawberry cake: $4.5
Cheesecake: $6.99
```
Exceptions

An exception is a disruptive event that occurs while a program is running typically indicates a runtime error

Examples: IndexOutOfBoundsException, NumberFormatException

When an error occurs, we throw the exception. Any function that is currently on the stack can catch the exception.

- Functions that do not catch the exception are aborted
- If no one catches the exception, the program terminates and prints the exception to the console
Exceptions are objects
Throwing an exception

```java
public static void bar() {
    throw new RuntimeException("An error happened in bar()");
}
```
Catching an exception

```java
try {
    bar();
} catch (RuntimeException e) {
    System.out.println("An exception occurred: "+e.getMessage());
    e.printStackTrace();
}
```
public static void bar() {
    throw new RuntimeException("ERROR");
}

public static void foo() {
    try {
        bar();
    } catch (RuntimeException e) {
        System.out.println("Exception: "+e.getMessage());
        e.printStackTrace();
    }
    System.out.println("Hello!");
}

public static void main(String[] args) {
    foo();
}
Exercise: Write a program that catches an ArrayOutOfBoundException
Exceptions: best practices

• A production-level application should never throw and uncaught exception
  • e.g. the user should never encounter an exception.
  • thrown exceptions are bugs

• Throwing an exception is meant to help the developer
  • Serious mistakes that will derail further execution of the program
  • Errors related to undefined behaviors typically throw exceptions
    • divide by zero
    • adding vectors with mis-matches sizes
    • out of array bounds
Exceptions: best practices

Exceptions are slow and should not be used for routine error checking

- For example, checking whether a user input an integer

```java
class CheckInteger {
    public static void main(String[] args) {

        int value = 0;
        boolean valid = false;
        while (!valid) {
            System.out.print("Enter an integer: ");
            String input = System.console().readLine();
            try {
                value = Integer.parseInt(input);
                valid = true;
            } catch (RuntimeException e) {
                System.out.println("Sorry, this value is invalid");
            }
        }

        System.out.println("You entered " + value);
    }
}
```
Exceptions: Trace this program

```java
int value = 0;
boolean valid = false;
while (!valid) {
    System.out.print("Enter an integer: ");
    String input = System.console().readLine();
    try {
        value = Integer.parseInt(input);
        valid = true;
    } catch (RuntimeException e) {
        System.out.println("ERROR");
    }
}
System.out.println("You entered "+value);
```