Announcements

- Final Exam on Dec 9
  - 12 pm – 3 pm, in SN 014 (this room)
  - Everything covered till Dec 5 (next Monday)
- Self-help exercises (section-wise) posted online
  - Solutions will not be posted till next Wednesday
- HW 6 description corrected on course web page
- HW 7 will be posted online by tonight
  - Due Wednesday of next week
  - Absolutely no late submission accepted
    - Solutions will be posted on Wednesday
- 2-min presentations by:
  - Brian Dawson
  - Erin Coughlin

Grading Approach

- Criteria:
  - HWs: 60%, mid-term: 10%, final: 25%, participation: 5%
- Will curve each and every individual HW / exam
  - Criteria: median score < 67 %
  - Already done for mid-term
    - Median was originally 54 %
  - Will also do for the final exam
    - If median score is less than 67%
  - Will also do for each homework (individually)
    - HW 1 median score: 100 %
    - HW 2 median score: 88 %
    - HW 3 median score: 95 %
    - HW 4 median score: 91 %
    - HW 5, HW 6, HW 7 --- will check and adjust (if needed)
What does this do?

- If c is a lower-case letter:
  ```java
  c = (char)((int)c - (int)'a' + (int)'A');
  ```
  - Converts:
    - 'a' -> 'A'
    - 'b' -> 'B'
    - '{' -> '['
    - '|' -> '\'
    - '}' -> ']'
  - Easier (and safer) way:
    ```java
    c = Character.toUpperCase(c);
    ```
  - For strings:
    ```java
    s = s.toUpperCase();
    ```

String comparison

```java
String s = "UNC";
String t = "UNC";
if (s == t)
    ...
String s = "UNC";
String t = "UNC";
if (s.equals(t)) ...
```

- Legal?
  - Yes, but will always be false! Why?
  - String is an object
    ```java
    s == t  : compares the reference values only!
    ```

- Instead:
  ```java
  if (s.equals(t)) ...
  ```

- Also: `s.compareTo(t)`
  - Returns negative integer if s < t (lexicographically)
  - Returns 0 if the strings are the same
  - Returns positive integer if s > t (lexicographically)
Arrays – 1 dimensional

- **Have fixed size, fixed type**
  - `int [] list = new int[10];`
    - `int [] list: creates reference`
      - Name: list
      - Type: reference to int array
      - Location: 1000
      - Value: undefined
    - `new int[10]:`
      - Creates 10-element int array
      - Returns location
    - `= : assignment`

- **Can also do:**
  - `int [] list = {1, 2, 3, 10, 20, 30};`
    - Sets size and type

Using arrays

- **list.length → 10**
  - **Note: length is a property → no () needed**
    ```java
    for (int i=0; i < list.length; i++)
        list[i] = i;
    
    boolean found = false;
    for (int i=0; i < list.length; i++)
        if (list[i] == key)
            found = true;
    ```
Subroutines (& parameters)

- public static void f(parameter list)

  public static void greet()
  {
    System.out.println("Hello");
  }
  public static void greet(String n)
  {
    System.out.println("Hello" + n);
  }
  public static void greet(String n, int age)
  {
    System.out.println("Hello" + n);
    System.out.println("You are " + age + " years old");
  }

  No return value

Parameters

- Must match in type, number, order
  - public static void f(int x, double y, String z)
    {
      ...
    }

  - Which of these are valid subroutine calls?
    - f(3, 4.5, "Hello"); ✓
    - f(3, 4, "Hello"); ✓ small violation
    - f(3.5, 4, "Hello"); ✗ type
    - f(3, 4.5); ✗ number
    - f("Hello", 3, 4.5); ✗ order
Returning a value

```java
public static int sum(int x, int y)
{
    return (x + y);
}
public static boolean close(double x, double y)
{
    return (Math.abs(x-y) < 0.001);
}
public static int search(int[] list, int key)
{
    for (int i = 0; i < list.length; i++)
        if (list[i] == key) return i;
    return -1;
}
```

- Usage ...

Overloading

```java
public static void greet()
{
    System.out.println("Hello");
}
public static void greet(String n)
{
    System.out.println("Hello" + n);
}
public static void greet(String n, int age)
{
    System.out.println("Hello" + n);
    System.out.println("You are " + age + " years old");
}
```

- Can repeat function name
  - Argument list used to match to function definition!