

# Garfield AP CS

## Strings

# Warm-up

- Write a method `compare3` that takes three String parameter and returns whether or not all three have the same text.

# Comparing strings

- Relational operators such as `<` and `==` fail on objects.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Barney") {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

- This code will compile, but it will not print the song.
- `==` compares objects by *references* (seen later), so it often gives `false` even when two `Strings` have the same letters.

# The equals method

- Objects are compared using a method named equals.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Barney")) {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

- This is a method that returns a value of type boolean

# String test methods

| Method                             | Description  |
|------------------------------------|--|
| <code>equals(str)</code>           | whether two strings contain the same characters                                |
| <code>equalsIgnoreCase(str)</code> | whether two strings contain the same characters, ignoring upper vs. lower case |
| <code>startsWith(str)</code>       | whether one contains other's characters at start                               |
| <code>endsWith(str)</code>         | whether one contains other's characters at end                                 |
| <code>contains(str)</code>         | whether the given string is found within this one                              |

```
String name = console.next();
if (name.startsWith("Dr.")) {
    System.out.println("Are you single?");
} else if (name.equalsIgnoreCase("LUMBERG")) {
    System.out.println("I need your TPS reports.");
}
```

# Strings question

- Write a program that reads a person's name and converts it into a "gangsta name."

Output (run 1):

```
Type your name, playa: Peter Griffin
```

```
(M)ale or (F)emale? m
```

```
Your gangsta name is "P. GRIFFIN Daddy Peter-izzle"
```

Output (run 2):

```
Type your name, playa: Marge Simpson
```

```
(M)ale or (F)emale? F
```

```
Your gangsta name is "M. SIMPSON Goddess Marge-izzle"
```

# Type char

- `char` : A primitive type representing single characters.
  - Each character inside a `String` is stored as a `char` value.
  - Literal `char` values are surrounded with apostrophe (single-quote) marks, such as `'a'` or `'4'` or `'\n'` or `'\''`
  - It is legal to have variables, parameters, returns of type `char`

```
char letter = 'S';  
System.out.println(letter);           // S
```
- `char` values can be concatenated with strings.

```
char initial = 'P';  
System.out.println(initial + " Diddy");  
// P Diddy
```

# The charAt method

- The chars in a String can be accessed using the charAt method.

```
String food = "cookie";  
char firstLetter = food.charAt(0);    // 'c'  
  
System.out.println(firstLetter + " is for " + food);  
System.out.println("That's good enough for me!");
```

- You can use a for loop to print or examine each character.

```
String major = "CSE";  
for (int i = 0; i < major.length(); i++) {  
    char c = major.charAt(i);  
    System.out.println(c);  
}
```

Output:

```
C  
S  
E
```

# char vs. int

- All `char` values are assigned numbers internally by the computer, called *ASCII* values.
  - Examples:
    - 'A' is 65,        'B' is 66,        ' ' is 32
    - 'a' is 97,        'b' is 98,        '\*' is 42
  - Mixing `char` and `int` causes automatic conversion to `int`.
    - 'a' + 10 is 107,        'A' + 'A' is 130
  - To convert an `int` into the equivalent `char`, type-cast it.
    - (char) ('a' + 2) is 'c'

# char vs. String

- "h" is a String  
'h' is a char (the two behave differently)

- String is an object; it contains methods

```
String s = "h";  
s = s.toUpperCase(); // "H"  
int len = s.length(); // 1  
char first = s.charAt(0); // 'H'
```

- char is primitive; you can't call methods on it

```
char c = 'h';  
c = c.toUpperCase(); // ERROR: "cannot be dereferenced"
```

- What is `s + 1`? What is `c + 1`?
- What is `s + s`? What is `c + c`?

# Comparing char values

- You can compare char values with relational operators:  
`'a' < 'b'` and `'X' == 'X'` and `'Q' != 'q'`

– An example that prints the alphabet:

```
for (char c = 'a'; c <= 'z'; c++) {  
    System.out.print(c);  
}
```

- You can test the value of a string's character:

```
String word = console.next();  
if (word.charAt(word.length() - 1) == 's') {  
    System.out.println(word + " is plural.");  
}
```

# String/char question

- A *Caesar cipher* is a simple encryption where a message is encoded by shifting each letter by a given amount.
  - e.g. with a shift of 3,  $A \rightarrow D$ ,  $H \rightarrow K$ ,  $X \rightarrow A$ , and  $Z \rightarrow C$
- Write a program that reads a message from the user and performs a Caesar cipher on its letters:

```
Your secret message: Brad thinks Angelina  
is cute  
Your secret key: 3  
The encoded message: eudg wklqnv dqjholqd  
lv fxwh
```