Review
Introduction to Computer Science I
CSE 1020

moodle.yorku.ca

equals versus ==

Question

What is the state of an object?

equals versus ==

Question

What is the state of an object?

Answer

The state of an object consists of the non-static attributes of the class and their values.

equals versus ==

Question

An object has an identity. This identity is unique. That is, two different objects have different identities. In more concrete terms, how do we think of an object's identity?

equals versus ==

Question

An object has an identity. This identity is unique. That is, two different objects have different identities. In more concrete terms, how do we think of an object's identity?

Answer

The address in memory where the object is stored.

When are two objects references the same?

What do we mean by the same?

- the same state

  ```
  Fraction sum = ...
  Fraction one = ...
  boolean same = sum.equals(one);
  ```

- the same identity

  ```
  Fraction sum = ...
  Fraction one = ...
  boolean identical = (sum == one);
  ```

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = new Fraction(2, 4);
```

What does `f.equals(g)` return?

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = new Fraction(2, 4);
```

What does `f.equals(g)` return?

Answer

```
true
```

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = new Fraction(2, 4);
```

What does `f == g` return?

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = new Fraction(2, 4);
```

What does `f == g` return?

Answer

```
false
```

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = f;
```

What does `f.equals(g)` return?

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = f;
```

What does `f.equals(g)` return?

Answer

```
true
```

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = f;
```

What does `f == g` return?

equals versus ==

Question

```
Fraction f = new Fraction(1, 2);
Fraction g = f;
```

What does `f == g` return?

Answer

```
true
```

Regular expressions

A regular expression allows us to express a pattern.

A detailed description of patterns can be found in the API of the `Pattern` class, which is part of the `java.util.regex` package.

Regular expressions

The class `String` contains the following methods.

```
public boolean matches(String regex)
```

tests whether this string matches the given regular expression.

```
public String replaceAll(String regex, String
replacement)
```

replaces each substring of this string that matches the given regular expression with the given replacement.

Regular expressions

Question

Which regular expression captures all 416 phone numbers?

Regular expressions

Question

Which regular expression captures all 416 phone numbers?

Answer

```
"416-\\d{3}-\\d{4}"
```

Regular expressions

```
output.print("Enter a 416 phone number: ");
String number = input.next();
String pattern = "416-\\d{3}-\\d{4}";
if (!number.matches(pattern)) {
   output.println("Not a 416 phone number!");
}
```

Regular expressions

```
File file = new File(name);
Scanner fileInput = new Scanner(file);
final String NUMBER = "\\d{16}";
final String MASK = "****************";
while (fileInput.hasNextLine())  {
  String line = fileInput.nextLine();
  line = line.replaceAll(NUMBER, MASK);
  output.println(line);
}
```

N

Write an app named `Test4B` that does the following.

- Prompt the user by printing

  ```
  Enter an integer >= 4:
  ```

  The integer should be entered on the same line. You may assume that the user always enters an integer.
- Read the integer entered by the user.
- As long as the user enters an integer smaller than 4, reprompt the user by printing

  ```
  Enter an integer >= 4:
  ```

  again and reading the integer entered by the user.
- Print an N of height `h`, where `h` is the integer greater than or equal to 4 entered by the user (see sample run on next slide).

N

Here is a sample run (where the user has entered -1, 0, 3 and 4)

```
Enter an integer >= 4: -1
Enter an integer >= 4: 0
Enter an integer >= 4: 3
Enter an integer >= 4: 4
*  *
** *
* **
*  *
```

Z

Write an app named `Test4D` that does the following.

- Prompt the user by printing

  ```
  Enter an integer >= 4:
  ```

  The integer should be entered on the same line. You may assume that the user always enters an integer.
- Read the integer entered by the user.
- As long as the user enters an integer smaller than 4, reprompt the user by printing

  ```
  Enter an integer >= 4:
  ```

  again and reading the integer entered by the user.
- Print an Z of height `h`, where `h` is the integer greater than or equal to 4 entered by the user (see sample run on next slide).

Z

Here is a sample run (where the user has entered -1, 0, 3 and 4)

```
Enter an integer >= 4: -1
Enter an integer >= 4: 0
Enter an integer >= 4: 3
Enter an integer >= 4: 4
++++
  +
 +
++++
```

Medal standing

Exercise

Make the app more robust by trying to correct typos. In particular, try to correct the situation where the user did not type one of the characters. For example, Canaa, Canad, etc should all be acceptable.

Eclipse's debugger

- Window $\longrightarrow$ Open Perspective $\longrightarrow$ Debug
- Determine the point in the code that you want to inspect
- Double click that point in the editor's marker bar
- Run $\longrightarrow$ Debug As $\longrightarrow$ Java Application
- On the Debug view's toolbar, click the Step Into button