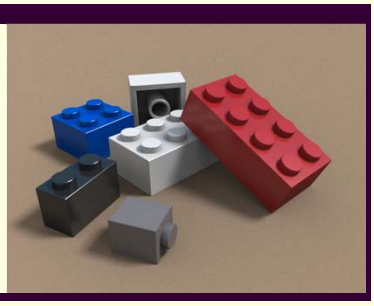
CS 152
Computer Programming Fundamentals
Class and Method Definitions

Instructor:
Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/



10/16/2017

Up Coming Schedule

Midterm Exam: *Friday, Oct 20*

Next lab assigned Friday, Oct 20 due: *Wed, Nov 1*

2

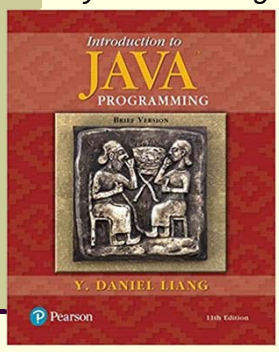
Midterm Exam:

- May bring one-page (back and front) of *hand written* notes.
- No calculators or other devices containing transistors.
- Questions like the i-clicker questions.
- Short answer - not multiple choice:
  - "This Java method compiles and runs. What is the output?"
  - "The Java code below contains a compile error. Which line and what is the error?"
  - "When the code below executes, a run-time error will occur. What error will be reported and on which line?"

3

Textbook & Reading for Midterm

Introduction to Java Programming, Brief Version (11th Edition)
By Daniel Liang



Midterm Exam Covers:

Chapter 1: Intro to Computers, Programs and Java

Chapter 2: Elementary Programming

Chapter 3: Selections

Chapter 4: Mathematical Functions, Characters and Strings

Chapter 5: Loops

Chapter 7: Arrays (7.1, 7.2, 7.3, 7.4)

Chapter 14: JavaFX (14.1, 14.3, 14.7, 14.11)

Chapter 15: Animations (15.11, 15.12)

4

Class and Method Definitions

```
1)  public class Dog
2)  {
3)    public String name;
4)    public int age;
5)
6)    public int getAgeInHumanYears()
7)    { int humanAge;
8)      if (age == 2) humanAge = age * 11;
9)      else humanAge = 22 + ((age-2) * 5);
10)     return humanAge;
11)   }
12)
13)   public static void main(String[] args)
14)   { Dog d1 = new Dog();
15)     Dog d2 = new Dog();
16)     d1.name = "Ralph";   d1.age = 8;
17)     d2.name = "Scooby";  d2.age = 2011-1969;
18)     System.out.println(d1.getAgeInHumanYears());
19)     System.out.println(d2.getAgeInHumanYears());
20)   }
21) }
```

Class Definition

Instance Variables

Method Definitions

Local Variables:
humanAge
d1, d2

Create object (instance of Dog)

Method Invocations

Class and Method Definitions

```
1)  public class Dog
2)  { public String name;
3)    public int age;
4)
5)    public int getAgeInHumanYears()
6)    { int humanAge;
7)      if (age == 2) humanAge = age * 11;
8)      else humanAge = 22 + ((age-2) * 5);
9)      return humanAge;
10)   }
11)
12)   public static void main(String[] args)
13)   { Dog d1 = new Dog();
14)     Dog d2 = new Dog();
15)     d1.name = "Ralph";   d1.age = 8;
16)     d2.name = "Scooby";  d2.age = 2011-1969;
17)     System.out.print(d1.name + " age=" + d1.age);
18)     System.out.println("==>" + d1.getAgeInHumanYears());
19)     System.out.print(d2.name + " age=" + d2.age);
20)     System.out.println("==>" + d2.getAgeInHumanYears());
21)   }
22) }
```

Output:

```
Ralph age=8==>52
Scooby age=42==>222
```

6

Java Application Structure

```
//imports

public class MyClass
{
  //public Class variables
  //private Class variables

  //method 1
  {
    //your code
  }

  public static void main(String[] args)
  {
    //your code
  }
}
```

7

Java Application Structure Example

```
1) import java.util.Scanner;
2) public class CircleCalculator
3) {
4)   private static final double PI = 3.14159265359;
5)
6)   public static void main(String[] args)
7)   {
8)     Scanner keyboard = new Scanner(System.in);
9)     double radius = keyboard.nextDouble();
10)    keyboard.close(); //Close Scanner when done using it.
11)    double area = findArea(radius);
12)    System.out.println("area=" + area);
13)  }
14)
15)  public static double findArea(double r)
16)  {
17)    return PI*r*r;
18)  }
19)}
```

CircleCalculator.findArea(4.6)

8

Quiz: Incorrect Use of a Local Variable

```
1)  public class Dog
2)  { private String name;
3)    private int age;
4)
5)    private int getAgeInHumanYears()
6)    {
7)      int humanAge;
8)      if (age == 2) humanAge = age * 11;
9)      else humanAge = 22 + ((age-2) * 5);
10)     return humanAge;
11)   }
12)
13)   public static void main(String[] args)
14)   {
15)     Dog d1 = new Dog();
16)     Dog d2 = new Dog();
17)     d1.name = "Ralph";
18)     d1.age = 8;
19)     System.out.println(d1.humanAge);
20)     System.out.println(d1.age);
21)   }
22) }
```

One line in this program contains a syntax error ☹

It attempts to use an ***out of scope local variable***.

The bad line is:

a) Line 17

b) Line 18

c) Line 19

d) Line 20

e) Line 21

9

static method: `isAllDigits`

```
1)  public class Bob
2)  {
3)    public static boolean isAllDigits(String str)
4)    {
5)      for (int i=0; i<str.length(); i++)
6)      {
7)        char c = str.charAt(i);
8)        if (!Character.isDigit(c)) return false;
9)      }
10)     return true;
11)   }
12)
13)   public static void main(String[] args)
14)   {
15)     System.out.println(isAllDigits("1492"));   //true
16)     System.out.println(isAllDigits("1492?"));  //false
17)     System.out.println(isAllDigits("3.141592654")); //false
18)     System.out.println(isAllDigits("00011811000")); //true
19)   }
20) }
```

The instant a non-digit is found, ***`return false`***!

The only way to reach this line is if line 8 never returned.

10

Main's a verses getBiggerArea's a

```
private static double getBiggerArea(double a, double b)
{
  if (a < b) a = b;
  return Math.PI * a*a;
}
```

Only *local copy* is changed.

The *values* passes from **main()** are copied into new **double** memory locations.

```
public static void main (String[] argv)
{
  double a = 1;
  double b = 10;
  double c = 2;
  System.out.println(getBiggerArea(a, b));
  System.out.println(getBiggerArea(a, c));
}
```

Output: 314.1592653589793
12.566370614359172

11

Quiz

```
public static int foo(int n)
{ System.out.print("foo(" + n + ")");
  if (n < 5) return 1;
  return -1;
}
public static void main(String[] args)
{ System.out.println("==" + "===>" + foo(2) );
  System.out.println("=====>" + foo(5) );
}
```

a)
```
foo(2)=====>1
foo(5)=====>-1
```

b)
```
=====>1 foo(2)
=====>-1 foo(5)
```

c)
```
=====>1
=====>-1
```

12

Overloaded Method: `sum()`

```
public static int sum(int a, int b, int c)
{ System.out.print("sum(int, int, int): ");
  return a+b+c;
}
public static int sum(double a, double b, double c)
{ System.out.print("sum(double, double, double): ");
  //return a+b+c;       //ERROR: Cannot convert double to int
  //return (int)a+b+c;  //ERROR: Cannot convert double to int
   //return (int)a + (int)b + (int)c;  //Ok, but different.
  return (int)(a+b+c);
}
public static void main(String[] args)
{ System.out.println(  sum(4,   6,   7  ) );
  System.out.println(  sum(4.4, 6.4, 7.4) );
}
```

```
sum(int, int, int): 17
sum(double, double, double): 18
```

13

What is Wrong with `foo()`

```
public static int foo(boolean cool)
{ if (cool == true)
   { return 1;
   }
   if (cool == false)
   { return -1;
   }
}
```

Compiler error Message:
"This method must return a result of type int."

... but, it does ...
... doesn't it? ...

```
public int foo(boolean cool)
{ if (cool == true)
   { return 1;
   }
   else
   { return -1;
   }
}
```

```
public int foo(boolean cool)
{ if (cool == true)
   { return 1;
   }
   return -1;
}
```

14

Quiz: Which Indenting is Correct?

```
public class Foo
{
  public int getY(int x)
  {
  int y = 2*x*x - 3*x;
  if (y < 0)
  {
    y += 5;
```

(a)

```
public class Foo
{
  public int getY(int x)
  {
    int y = 2*x*x - 3*x;
    if (y < 0)
    {
      y += 5;
```

(b)

```
public class Foo
  {
  public int getY(int x)
    {
    int y = 2*x*x - 3*x;
    if (y < 0)
      {
        y += 5;
```

(c)

```
public class Foo
{
  public int getY(int x)
  {
  int y = 2*x*x - 3*x;
    if (y < 0)
    {
    y += 5;
```

(d)

15

Method: `badSwap(int a, int b)`

```
public class HelloWorld
{ static void badSwap(int a, int b)
  { int temp = a;
    a = b;
    b = temp;
  }
  public static void main(String[] args)
  { int a = 5;
    int b = 7;
    System.out.println("a="+a + ", b="+b);
    badSwap(a, b);
    System.out.println("a="+a + ", b="+b);
  }
}
```

Output:

```
a=5, b=7
a=5, b=7
```

16

JavaDoc: `java.awt.Point`

Fields x and y are public: no need to use get() / set().

Field Summary

int	**x** — The *x* coordinate.
int	**y** — The *y* coordinate.

Constructor Summary

Point() — Constructs and initializes a point at the origin (0, 0) of the coordinate space.
Point(int x, int y) — Constructs and initializes a point at the specified (*x*, *y*) location in the coordinate space.
Point(Point p) — Constructs and initializes a point with the same location as the specified Point object.

Method: `swap(Point p)`

```
static void swap(Point p)
{ int temp = p.x;
  p.x = p.y;
  p.y = temp;
}
```

In Java, all non-primitive types (objects) are ***passed by reference.***

```
public static void main(String[] args)
{ Point myPoint = new Point(5,7);
  System.out.println(
                "a="+myPoint.x + ", b="+myPoint.y);
   swap(myPoint);
  System.out.println(
                "a="+myPoint.x + ", b="+myPoint.y);
}
```

Output:

```
a=5, b=7
a=7, b=5
```

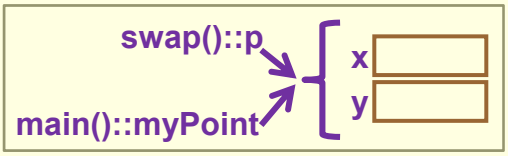
18

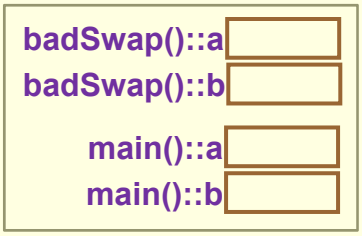
Method: `swap(Point p)`

```
static void swap(Point p)
{ int temp = p.x;
  p.x = p.y;
  p.y = temp;
}

static void badSwap(int a, int b)
{ int temp = a;
  a = b;
  b = temp;
}

public static void main(String[] args)
{ Point myPoint = new Point(5,7);
  int a = 5, b = 7;
  swap(myPoint);   badSwap(a, b);
}
```





19

Quiz: What is the output?

```
1)  private static int abs(int x)
2)  {
3)    if (x < 0) x = -x;
4)    System.out.print(x + " ");
5)    return x;
6)  }
7)
8)  public static void main(String[] args)
9)  {
10)   int x = -2;
11)   int y = -5;
12)   int a = abs(x);
13)   int b = abs(x+y);
14)   System.out.print(x + " ");
15)   System.out.print(y + " ");
16)   System.out.print(a + " ");
17)   System.out.print(b + " ");
18) }
```

a) 2 7 2 5 2 7
b) 2 7 -2 -5 2 7
c) 2 3 -2 -5 2 3
d) 2 3 2 5 2 3
e) 2 3 2 -5 2 3

20

What is the output?

```
1)  private static int abs(int x)
2)  {
3)    if (x < 0) x = -x;
4)    System.out.print(x + " ");
5)    return x;
6)  }
7)
8)  public static void main(String[] args)
9)  {
10)   int x = -2;
11)   int y = -5;
12)   int a = abs(x);    //2
13)   int b = abs(x+y);  //7
14)   System.out.print(x + " ");  //-2
15)   System.out.print(y + " ");  //-5
16)   System.out.print(a + " ");  //2
17)   System.out.print(b + " ");  //7
18) }
```

b) 2 7 -2 -5 2 7

21

X Raised to the Yth Power: **all in `main()`**

```
1) public class HelloWorld
2) { public static void main(String[] args)
3)   {
4)     int x=2, y=4, pow = 1;  // Declared & Set Values
5)     for (int i=0; i<y; i++)
6)     { pow *= x;
7)     }
8)     System.out.println(pow);    //output: 16
9)
10)    x=3; y=4; pow = 1;      // Set Values
11)    for (int i=0; i<y; i++)
12)    { pow *= x; //pow = pow * x;
13)    }
14)    System.out.println(pow);  //output: 81
15)  }
16)}
```

22

X Raised to the Yth Power: **Method**

```
1)  public class HelloWorld
2)  {
3)    private static int xToPowerY(int x, int y)
4)    { int pow = 1;
5)      for (int i=0; i<y; i++)
6)      { pow *= x;
7)      }
8)      return pow;
9)    }
10)
11)   public static void main(String[] args)
12)   {
13)     System.out.println(xToPowerY(2,4)); //16
14)     System.out.println(xToPowerY(3,4)); //81
15)   }
16) }
```

23

Quiz: Method return

```
1)  public static int foo(boolean cool)
2)  { System.out.print("1 ");
3)    if (cool == true)
4)    { System.out.print("2 ");
5)      return 2;
6)    }
7)    else if (cool == false)
8)    { System.out.print("3 ");
9)      return 3;
10)   }
11)   System.out.print("4 ");
12)   return 4;
13) }
14) public static void main(String[] args)
15) { System.out.println("["+foo(false)+"]");
16) }
```

a) 1 3 [3]
b) 1 2 3 [3]
c) 1 3 3 [3]
d) 1 3 4 [3]
e) 1 3 4 [4]

24

What is an Algorithm for Determining Primality?

- A natural number is ***a prime number*** if it is greater than one and has no divisors other than 1 and itself.
- 1 is not a prime number.
- The first 25 primes: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.
- How would a human go about determining if a given number is prime?
- Is 2372 prime?
- Is 3333 prime?
- What is different about how a computer would answer this question?

25

An Algorithm for Determining Primality

- An algorithm for determining whether a number, n, is a prime:
  - Create a loop that starts at n-1 and counts down in whole numbers to 2.
  - Inside the loop, divide n by the loop index. If on any iteration of the loop, the remainder equals zero then n is not prime.
  - If the loop finishes without a single divisor having been found, then n is prime.

26

isPrime(int n)

How could this be more efficient?

```
1)  public static boolean isPrime(int n)
2)  { boolean prime = true;
3)    for (int i=n-1; i>1; i--)
4)    { System.out.print(i + " ");
5)      if (n % i == 0) prime=false;
6)    }
7)    return prime;
8)  }
9)
10) public static void main(String[] args)
11) { System.out.println(isPrime(10));
12)   System.out.println(isPrime(11));
13)   System.out.println(isPrime(15));
14) }
```

A natural number is ***a prime number*** if it is greater than one and has no divisors other than itself and 1.

```
9 8 7 6 5 4 3 2 false
10 9 8 7 6 5 4 3 2 true
14 13 12 11 10 9 8 7 6 5 4 3 2 false
```

27

Return As Soon As a Factor is Found

```
1)  public static boolean isPrime(int n)
2)  { for (int i=n-1; i>1; i--)
3)    { System.out.print(i + " ");
4)      if (n % i == 0) return false;
5)    }
6)    return true;
7)  }
8)
9)  public static void main(String[] args)
10) {
11)   System.out.println(isPrime(10));
12)   System.out.println(isPrime(11));
13)   System.out.println(isPrime(15));
14) }
```

```
9 8 7 6 5 false
10 9 8 7 6 5 4 3 2 true
14 13 12 11 10 9 8 7 6 5 false
```

28

Small Primes are more Densely Spaced

```
1)  public static boolean isPrime(int n)
2)  { for (int i=2; i<n; i++)
3)    { System.out.print(i + " ");
4)      if (n % i == 0) return false;
5)    }
6)    return true;
7)  }
8)  public static void main(String[] args)
9)  {
10)   System.out.println(isPrime(10));
11)   System.out.println(isPrime(11));
12)   System.out.println(isPrime(15));
13) }
```

```
2 false
2 3 4 5 6 7 8 9 10 true
2 3 false
```

29

How Can This Be Improved?

```
1)  public static void main(String[] args)
2)  { System.out.println(isPrime(163));
3)  }
4)  public static boolean isPrime(int n)
5)  { for (int i=2; i<n; i++)
6)    { System.out.print(i + " ");
7)      if (n % i == 0) return false;
8)    }
9)    return true;
10) }
```

What if this were 7919 or 152,953,277?

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106
107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 true
```

30

Algorithm for Greatest Common Factor

- The GCF of 30 and 70 is 10.
- The GCF of 30 and 60 is 30.
- 1000 (2*2*5*5) and 81 (3*3*3*3) are coprime.

Given a pair of integers, how could the GCF be found?

Given a pair of positive integers: a, b.

If b is less than a, then swap.

Starting with a, check each integer down through 2.

The first integer found that divides both a and b is the GCF.

If no such integer is found, then the two integers are coprime.

31

getGCF(int a, int b)

```
1) public class Explore
2) { public static void main(String[] args)
3)   { System.out.println( getGCF(  30, 70) );
4)     System.out.println( getGCF(1000, 81) );
5)   }
6)
7)   public static int getGCF(int a, int b)
8)   { if (a > b)
9)     { int tmp = a;    a=b;    b=tmp;  //swap
10)    }
11)    for (int i=a; i>=2; i--)
12)    { if ((a % i == 0) && (b % i == 0))
13)      { return i;
14)      }
15)    }
16)    return 1;
17)  }
18)}
```

32

Quiz: Information Hiding

In the context of computer programming, Information Hiding is used to:

a) Increase security.

b) Make it easier to use a method.

c) Increase the method's efficiency.

d) Decrease the method's efficiency.

e) Reduce the level of abstraction.

33