Essential Question:

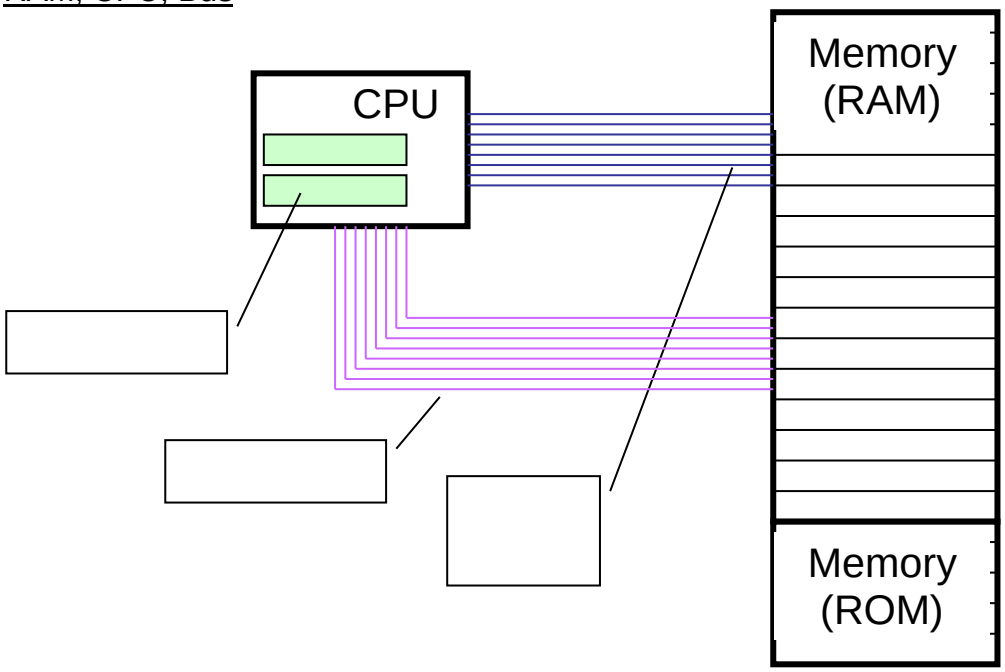
QUESTIONS	NOTES:

The text for this chapter is on the web at: www.skylit.com/javamethods

Hardware

- The CPU (_____) is made of millions of _____ devices, called _____, etched into a silicon chip.
- Transistors are combined to form _____.
- All _____ is basically made up of gates.

RAM, CPU, Bus

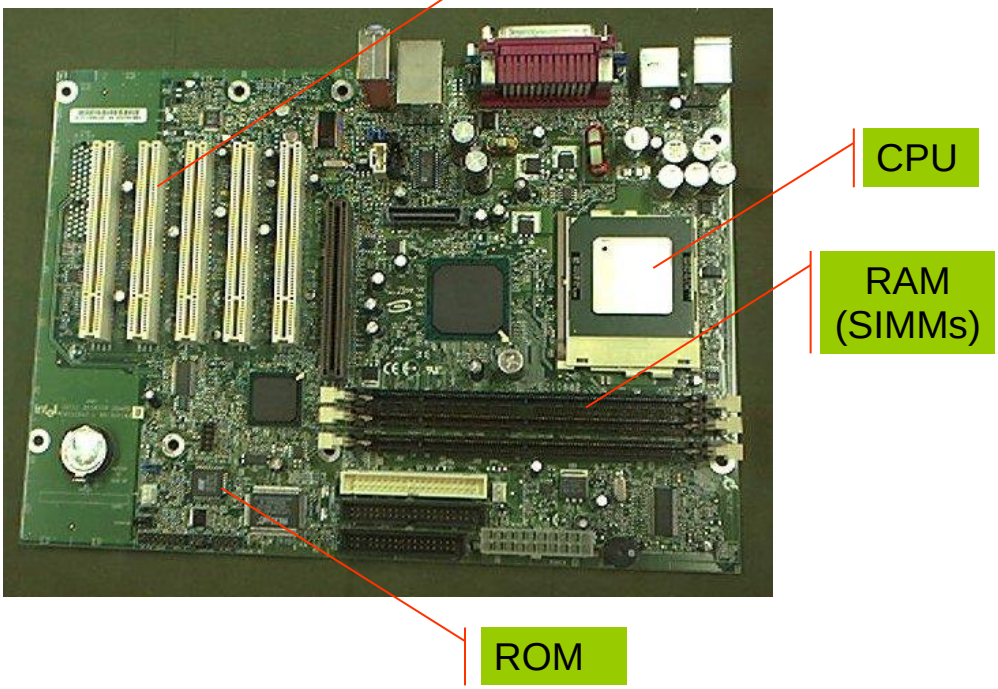


Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

Motherboard



Hardware Terms

- CPU — _____
- RAM — _____
  - "random-access" means the CPU can read directly from and write to any memory location
  - holds both data and CPU instructions
- ROM — _____
  - holds initialization and hardware diagnostic programs

Summary/Reflection:

QUESTIONS	**NOTES:**

CPU

- In personal computers, the CPU is a _____
- The CPU speed is measured in MHz (_____, millions of clock cycles per second) and GHz (_____, billions of cycles).
- A CPU _____ takes one or several clock cycles.

RAM

- 1 KB (kilobyte) = _____ bytes
- 1 MB (megabyte) = _____ bytes
- 1 GB (gigabyte) = _____ bytes

Input/Output (I/O) Devices

- _____ and video adapter
- _____, _____ or touch pad
- Sound card, _____, _____
- Internet adapter, _____
- D/A (digital-to-analog) and A/D (analog-to-digital) converters
- Scanners, _____, printers

Summary/Reflection:

QUESTIONS	**NOTES:**

Software Terms

- Operating system
  - a program that _____, _____, and provides other _____
- Console application
  - a program with _____
- GUI — _____
  - graphics, menus, buttons, icons, etc.
- OOP — _____

Software Developers Have To:

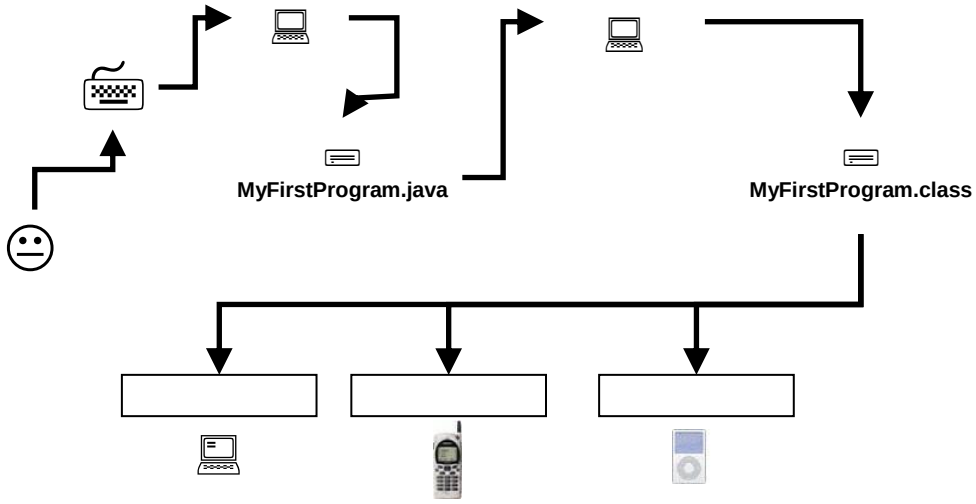
- Absorb and use _____ technical information
- Create sound software system _____
- Understand and devise effective _____
- Be proficient with the _____ of programming languages
- Diagnose and correct programming errors (_____)
- Use _____ and documentation
- Find and utilize _____ software components
- Design and implement _____ user interfaces
- Uphold the highest standards of _____

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Java's Compiler + Interpreter



Structure of a Java Program

For Unit 2, you will be writing the most basic form of Java Programs. They look like this:

```
public class SomeName
{
    public static void main (String [ ] args)
    {
        statement;
        statement;
        statement;
                                                    …
    }
}
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Documentation
(called "comments" in Computer Science)

```
// Name:
// Last Changed:
// Description: What does this program do.
public class SomeName
{
    public static void main (String [ ] args)
    {
        statement;
                                                                    …
    }
}
```

Types of Errors

Syntax Error:

Logic Error:

Output to the console screen in Java
Two output methods:

1) System.out.println(someString);
2) System.out.print(someString);

Examples

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

OOP — Object-Oriented Programming

- An OOP program models a _____.
- An object may have its own "_____," which may contain other _____.
- An object has a set of _____ that can process _____ of certain types.
- A method can _____, _____, and _____.
- An object belongs to a particular _____, and the functionality of each object is determined by its class.
- A programmer creates an OOP application by _____.

The Main OOP Concepts:

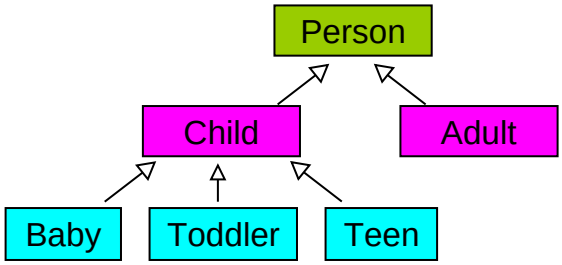
- Inheritance: a *subclass* _____ a *superclass*; the objects of a subclass _____ of the superclass and can _____ them or add ___ features.
- Event-driven programs: the program simulates _____ handling of events; methods are called automatically in _____.

Summary/Reflection:

QUESTIONS	**NOTES:**

Inheritance

- A programmer can define hierarchies of _____
- More general classes are closer to the _____



Ex. Using the Turtle class, create a Turtle object and "command" that Turtle object to draw a hexagon with side length 50 pixels.

```
import java.awt.Color;

public class Hexagon
{
    public static void main(String [ ] args)
    {
        Turtle t;
        t = new Turtle();
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Get it, Got it, Good to go!

Using the Turtle class, create a Turtle object and "command" that Turtle object to draw an equilateral triangle with side length 100 pixels.

```
import java.awt.Color;

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

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

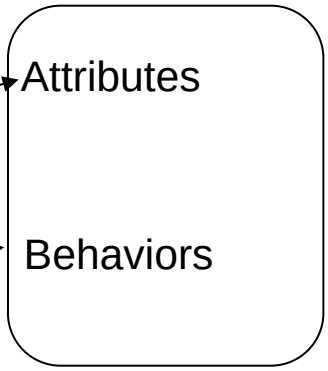
Classes and Objects

- A *class* is a piece of the program's source code that _____. OO programmers write _____.
- An _____ is called an _____. A program can create and use more than one object (instance) of the same class.

Class

- A _____ for objects of a particular type
- _____ (number, types) of the attributes
- Defines _____ _____ of its _____

Object

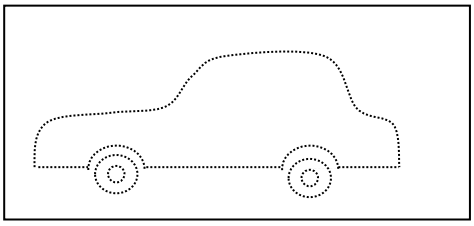


Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

Class: Car



Attributes:
- String model
- Color color
- int numPassengers
- double amountOfGas

Behaviors:
- Add/remove a passenger
- Get the tank filled
- Report when out of gas

Object: a car

Attributes:
- model = "Mustang"
- color = Color.YELLOW
- numPassengers = 0
- amountOfGas = 16.5

Behaviors:

Summary/Reflection:

QUESTIONS	**NOTES:**

Class VS Object

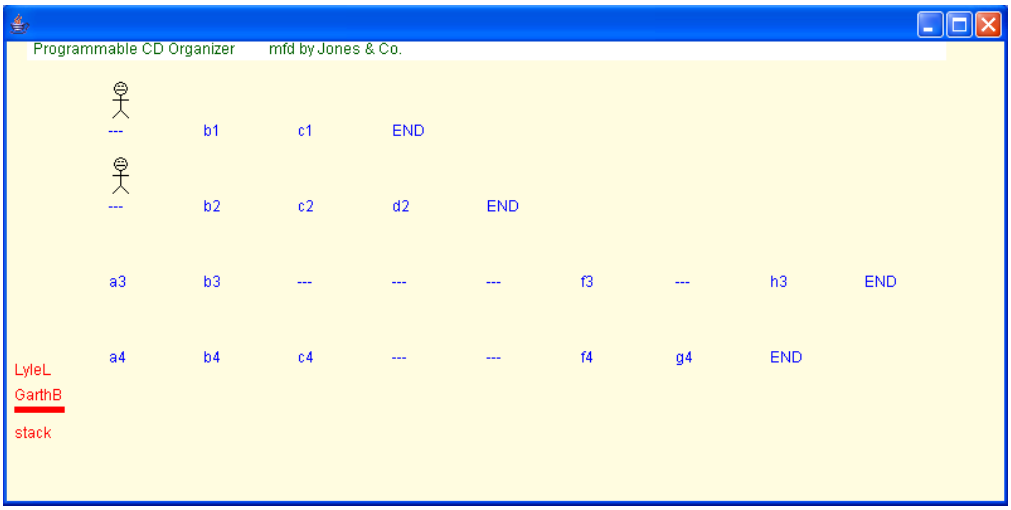
_____ _____ _____	An _____ in a _____ program
Written by a _____	Created when the program is running (by the _____ or a _____ or another _____)
_____ (the number and types) of its objects' attributes — the _____ for all of its objects	Holds _____ of attributes; these can _____ while the program is _____
Specifies the possible _____ of its objects	Behaves appropriately when _____

Summary/Reflection:

QUESTIONS	NOTES:

Introducing The Vic Class

A programmable machine that stores CD's and moves them around with operator commands.



Summary/Reflection:

QUESTIONS	**NOTES:**

Ex. Write an application program that creates a Vic object and then swaps the CD in its third slot with the CD in its second slot. Assume it has CDs in both slots.

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

Summary/Reflection:

QUESTIONS	NOTES:

Get it, Got it, Good to Go!

Write an application program that creates a Vic object and then picks up a CD in the 4th slot and puts it in the 2nd slot. Assume there is a CDS in the 4th slot and there is no CD in the 2nd slot.

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

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Variables

- A variable is a "named _____" that holds a _____.
- Example: q = 100 - q;
  means:
  - 1. Read the current value of ___
  - 2. Subtract it ____ 100
  - 3. _____ the result back into q

Fundamental Data Types in Java

Type Name	Kind of data	Memory used	Size range
		4 bytes	-2,147,483,648 to 2,147,483,647
		8 bytes	±1.76769 x 10^308 to ± 4.94 x 10^-324
		2 bytes	all characters
		1 bit	true or false

ALL OTHER TYPES ARE CLASSES!!!

Your first Java class

Type Name	Kind of data	Memory used	Size range
	words or phrases	varies	any word or phrase

Summary/Reflection:

QUESTIONS	**NOTES:**

A _____ must be _____ before it can be used:

```
int    count;
double    x, y;
JButton    go;
Walker    amy;
String   firstName;
```

The _____ = _____ the variable's value:

Examples

```
int numberOfStudents;

double myGPA = 4.15;

boolean studentAbsent = false;

String firstName;

String phoneNumber = "707-433-5777";
```

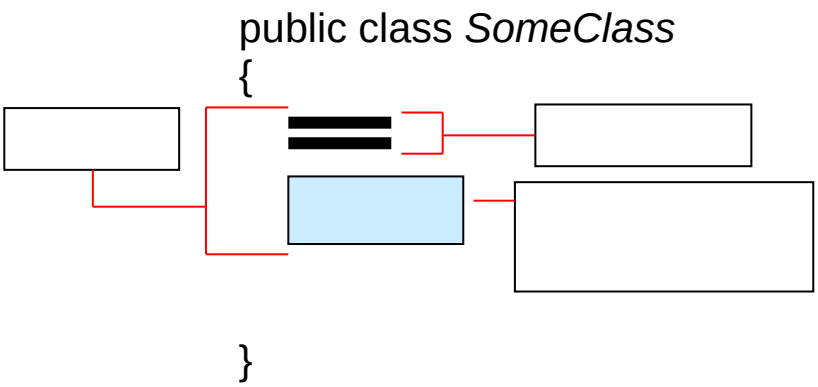
Summary/Reflection:

QUESTIONS	**NOTES:**

Variables: Scope

- Each variable has a _____ — the area in the source code where it is "_____."
- If you use a variable _____ its scope, the compiler reports a _____ error.
- Variables can have the same _____ when their scopes do _____ overlap.

Variables: Fields



Variables: Local Variables

- Local variables are declared _____ a constructor or a method.
- Local variables _____ their values and are _____ once the constructor or the method is _____.
- The _____ of a local variable is from its _____ down to the closing _____ of the block in which it is declared.

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Math Stuff in Java

Arithmetic Operators

Shortcuts in Java

```
        means
        means
        means
        means
        means
        means
```

Summary/Reflection:

QUESTIONS	**NOTES:**

More Math Stuff (need: import java.lang.Math;)

Examples

```
double a, b, c, d;
a = 100;
b = 2
c = -12.6;
d = Math.sqrt(a);
e = Math.pow(d, b);
```

Ex. Using the template provided, complete the methods of the **Circle** class.

Get it, Got it, Good to Go!

Using the template provided, complete the methods of the **Rectangle** class.

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Conditional Statements in Java
(also called "decision" statements)

We have 3 ways of making decisions:

1. _____

Syntax:

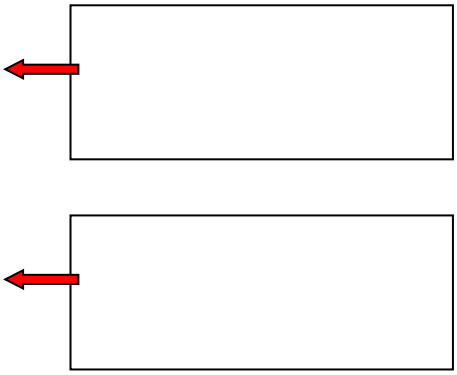
```
if(condition)
{
     statement;
     statement;
     …
}
```

QUESTIONS	**NOTES:**

2. _____

Syntax:

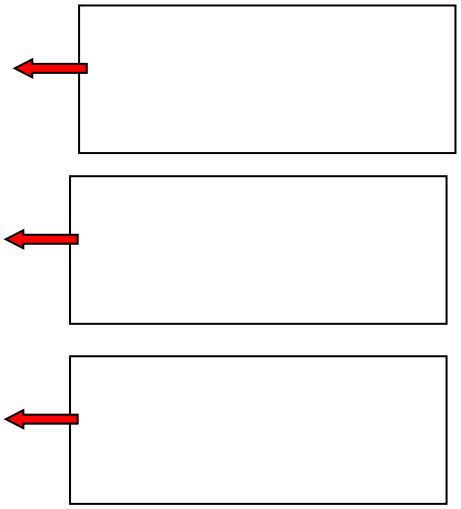
```
if(condition)
{
      statement;
      statement;
      …
}
else
{
      statement;
      statement;
      …
}
```



3. _____

Syntax:

```
if(condition1)
{
      statement;
      …
}
else if(condition2)
{
      statement;
      …
}
else
{
      statement;
      …
}
```



Summary/Reflection:

QUESTIONS | **NOTES:**

Logical Operators

At times we need to ask more complex questions:

- if more than 1 condition is true (AND)
- if either condition is true (OR)
- if a condition is not true (NOT)

Logical operators in Java (and in C and C++)

Logic Java operator

Truth Tables in Logic

p	q	p and q	p or q	not p	not q
T	T				
T	F				
F	T				
F	F				

Summary/Reflection:

QUESTIONS	**NOTES:**

Relational Operators

Relational operators in Java (and in C and C++)

Math	Java operator	Means

Order of Operations

1	2	3	4	5	6	7	8	9	10	11

Summary/Reflection:

QUESTIONS	**NOTES:**

Example

Given 2 int values, return true if either of them is in the range 10..20 inclusive.

```
public boolean in1020(int a, int b)
{
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Example

Given 2 int values, return whichever value is nearest to the value 10, or return 0 in the event of a tie. Note that Math.abs(n) returns the absolute value of a number.

```
public int close10(int a, int b)
{









}
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Vic Example

Write an application that creates two Vic objects and performs the following task:

For each slot that both Vics have valid slots, do the following:

- if both Vics have CDs, just move both Vics to the next slot

- if only one of the Vics has a CD in its slot, take the CD and put it in the other Vics slot, and move both Vics to the next slot

- if neither of the Vics have CDs, just move both Vics to the next slot

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

Essential Question:

QUESTIONS	NOTES:

Short-Circuit Evaluation
if (*condition1* **&&** *condition2*) ...

If *condition1* is ____, then _____ is _____
(the result is _____ anyway)
_____

if (*condition1* **||** *condition2*) ...

If *condition1* is ____, then _____ is _____
(the result is _____ anyway)

De Morgan's Laws

One could think of it as the "_____" for Logic

! (p **&&** q) evaluates to _____
! (p **||** q) evaluates to _____

Examples of DeMorgan's Law:

Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS | **NOTES:**

The `switch` Statement

switch
case
default
break

Reserved words

Only works with:

```
switch
    (expression)
{
    case value1:
       ...
     break;

    case value2:
       ...
     break;
    ...
    ...
    default:
     ...
     break;
}
```

Don't forget **break**s !

Summary/Reflection:

QUESTIONS	**NOTES:**

The Cast Operator

- Java allows a programmer to "_____" a piece of data to a _____ type when needed.
- Can be useful if working with int values and a decimal result is needed. For example, one might want the decimal equivalent of a fraction even though the numerator and denominator are integers.

Example:
decAnswer =

Example: Use Casting to round a number to the nearest ones digit.

_____

Math.Random()

- returns a _____ value _____
- Can then be _____ to fit the needs of your application.
- For example, if you needed a random number from 1 to 12, you would do something like this:

randNum = _____**;**

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Loops in Java

A structure that allows the programmer to _____ something while a condition is true. (also called "_____")

Three kinds of loops in Java

1. The _____ loop // we will see today
2. The _____ loop // we will see next lesson
3. The _____ loop // not on AP exam…in your book

1. The "while" loop – used when we want to repeat something while a condition remains true.

Syntax:

```
while(condition)
{
     statement;
     statement;
     …
}
```

Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

Example:

```
// Returns the sum of the first
//  n integers
public int sumAll (int n)
{
   int p = 1;
   int sum = 0;
   while ( p <= n )
   {
      sum += p;
      p++;
   }
   return sum;
}
```

- _____: The variables tested in the condition must be initialized to some values. If the condition is _____ at the outset, the loop is _____ entered.
- _____: The condition is tested before each _____. If false, the program continues with the first statement _____ the loop.
- _____: At least _____ tested in the condition must change within the _____ of the loop.

Summary/Reflection:

QUESTIONS	NOTES:

Extending the Vic

Shortcoming: Vic does not have a method to go back to the front of the sequence (the first slot).

Extension: called _____ in OOP. Allows us to keep all of the qualities of a class, plus add any new ones.

Introducing the DualDirectionVic

Allows us to keep all of the qualities of a Vic, plus add any new ones.

The new ones:

1. _____
2. _____
3. _____

```
class DualDirectionVic extends Vic
{
    private String itsFirstPosition;

    public DualDirectionVic() {
        super();
        itsFirstPosition = getPosition();
    }

    public void goToFirst() {
        while(!seesFirstSlot())
           backUp();
    }

    public void goToLast() {
        while(seesSlot())
            moveOn();
        backUp();
    }

    public boolean seesFirstSlot() {
        if(itsFirstPosition.equals(getPosition()))
            return true;
        else
            return false;
    }
}
```

QUESTIONS	**NOTES:**

Example

Given one Vic object. Return the number of CDs that are in its row.

```
public int numCDs(Vic v)
{









}
```

QUESTIONS	**NOTES:**

Get it, Got it, Good to Go!

Given two Vic objects. Return the absolute difference between the number of CDs in v1 and v2

```
public int absDifference(Vic v1, Vic v2)
{









}
```

Essential Question:

QUESTIONS	NOTES:

The for Loop

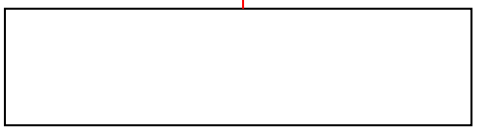
- **for** is a loop that combines in one statement _____, _____, and _____.

```
for ( _____;_____;_____ )
{
   statement1;
   statement2;
   ...
   statementN;
}
```

- **for** loops are great for _____ something a certain amount of times

- Java allows you to declare the loop control variable in the **for** statement itself. For example:

```
for (int i = 0;  i < n ;  i++)
{
    ...
}
```



Summary/Reflection:

QUESTIONS	**NOTES:**

"Repeat *n* times" idiom:

```
for (int i = ____;  _________________ ;  i++)
{
    ...
}
```

or

```
for (int count = ____;  _________________ ;  count++)
{
    ...
}
```

Equivalent while loop below:

Summary/Reflection:

QUESTIONS	**NOTES:**

Example:

```
// Returns the sum of the first
// n integers
public int sumAll (int n)
{
   int sum = 0;
   for (int p = 1;  p <=n;  p++)
   {
      sum += p;
   }
   return sum;
}
```

Example: Write the method to compute n! (factorial)

```
public int factorial (int  n)
 {




 }
```

Summary/Reflection:

QUESTIONS	**NOTES:**

The Role of <u>break</u> and <u>return</u> in Loops

- _____ in a loop instructs the program to _____ quit the current _____ and go to the first statement _____ the loop.
- _____ in a loop instructs the program to immediately quit the current _____ and return to the calling method.
- A break or return must be _____ an ___ or an _____, otherwise the code after it in the body of the loop will be _____.

<u>Example:</u>

```
    int d = n – 1;

while (d  > 0)
{
   if (n % d == 0)
      break;
   d––;
}

if ( d > 0 )   // if found a divisor

          ...
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Nested Loops

A _____ is called nested.

Example

```
    // Draw a 5 by 3 grid:

for (int x = 0;  x < 50;  x += 10)
{
    for (int y = 0;  y < 30;  y += 10)
    {
        g.fillRect(x, y, 8, 8);
    }
}
```

Draw a sketch of what is drawn below.

Summary/Reflection:

Essential Question:

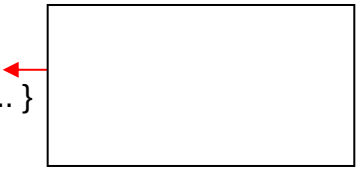
QUESTIONS	NOTES:

Encapsulation

- A class (in OOP) _____ (groups together) _____ and _____ relating to an object.
- Encapsulation _____ _____ of the class from the users (clients) of the class.

```
public class MyClass
{
    // Private fields:
    private  <sometype>  myField;
    ...

    // Constructors:
    public MyClass (...)   { ... }
    ...
    // Public methods:
    public  <sometype>  myMethod (...)   { ... }
    ...
    // Private methods:
    private  <sometype>  myMethod (...)  { ... }
    ...
}
```



Summary/Reflection:

QUESTIONS	**NOTES:**

1. _____ (or instance variables)

Fields describe the _____ (*think nouns!!*).

Fields are generally labeled _____ and therefore hidden from the client (only accessible from _____ the class).

2._____

- Constructors describe ways to _____ _____ of a class and _____.
- Constructors are always labeled _____ and have the _____ as the class.
- There may be _____ constructor for a class. If more than one, each must have different _____ (or _____).

Summary/Reflection:

QUESTIONS	**NOTES:**

3. _____

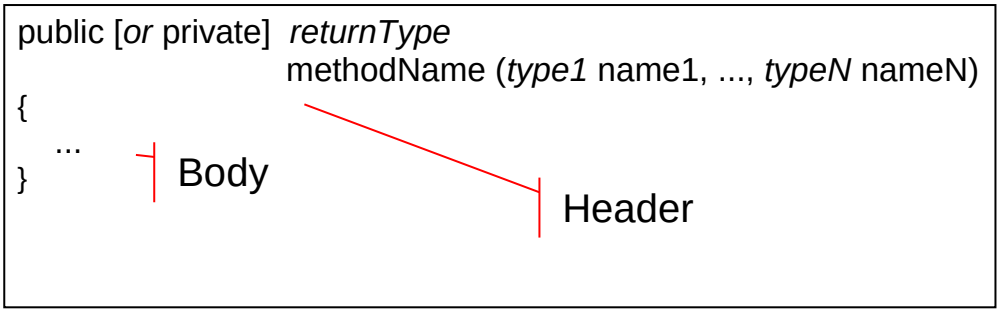
- Methods describe the _____ or _____ that can be asked about an object (*think verbs!!*).
- Methods are _____ labeled _____. private (_____) methods can only be used _____ the class.

In general, 3 classifications of methods:

1. _____ – "gets" information about the object.
2. _____ – "sets" information about the object (or changes the state of the object).
3. _____ – asks a true or false question about the object.

Summary/Reflection:

QUESTIONS	**NOTES:**



Steps To define a method:

1.

2.

3.

4.

5.

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Passing Parameters to Constructors and Methods

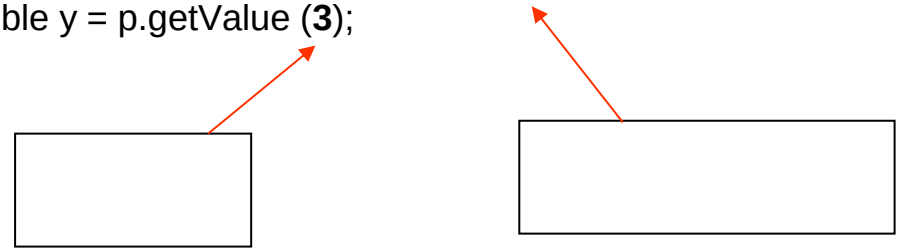
- Any expression that has an _____ can serve as a parameter:

```
double u = 3, v = -4;
...
Polynomial p = new Polynomial(1.0,  -(u + v),  u * v);
double y = p.getValue(2 * v - u);
```

```
public class Polynomial
{
   public Polynomial (double a, double b, double c) { ... }
   public double getValue (double x) { ... }
   ...
}
```

- _____ is promoted to _____ when necessary:

...
Polynomial p = new Polynomial (**1, –5, 6**);
double y = p.getValue (**3**);

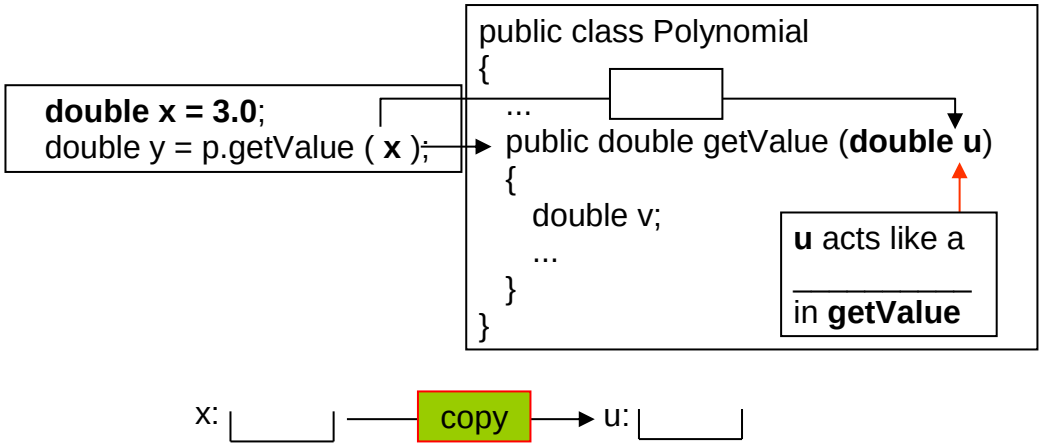


Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

Primitive data types are always passed "_____": the value is _____ into the parameter.



Summary/Reflection:

QUESTIONS	**NOTES:**

```
public class Test
{
   public double square (double x)
   {
      x *= x;
      return x;
   }

   public static void main(String[ ] args)
   {
      Test calc = new Test ();
      double x = 3.0;
      double y = calc.square (x);
      System.out.println (x + "  " + y);
   }
}
```

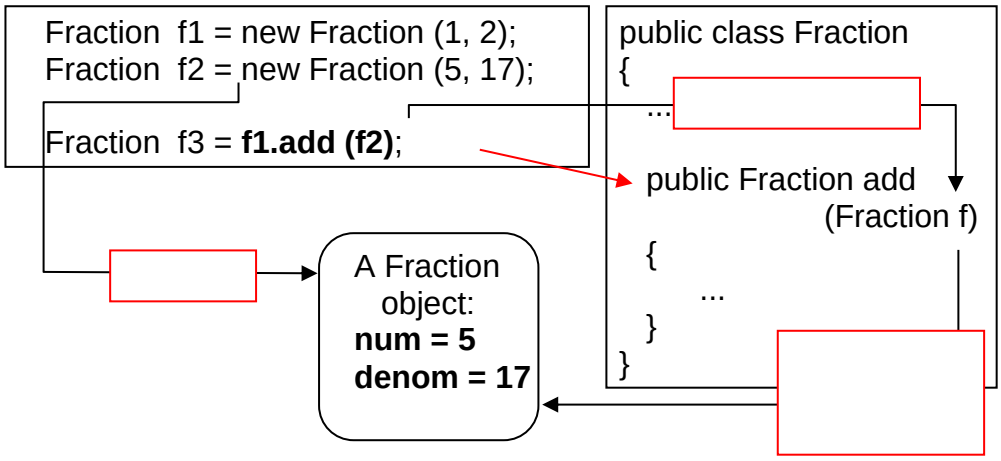
x here is a _____ of the parameter passed to **square**. The copy is _____, but...

... the _____ **x** is _____. Output: _____

Summary/Reflection:

QUESTIONS | **NOTES:**

_____ are always passed as _____:
the _____ is copied, not the object.



- A method can _____ passed to it as a parameter (because the method gets a _____ to the _____ object).
- A method can change the object for _____ (this object acts like an implicit parameter):
- Inside a method, _____ refers to the object for which the method was called. this can be passed to other _____ and _____ as a _____:

Summary/Reflection:

QUESTIONS	**NOTES:**

Example: Suppose the method fun is defined as:

```
public int fun(int x, int y)
{
     y -= x;
     return y;
}
```

What is printed to the screen after the following code is executed?

```
int a = 3, b = 7;
b = fun(a, b);
a = fun(b, a);
System.out.println(a + " " + b);
```

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

The return Statement

- A method, _____, returns a value of the _____ to the calling method.
- The return statement is used to _____ quit the method and return _____:

```
return expression;
```

The _____ of the return value or expression must _____ the method's declared return type.

_____

A method can have _____ return statements; then _____ of them must be inside an if or else (or in a switch):

```
public someType myMethod (...)
{
   ...
   if (...)
      return <expression1>;
   else if (...)
      return <expression2>;
   ...
   return <expression3>;
}
```

Summary/Reflection:

QUESTIONS	**NOTES:**

A boolean method can return _____, _____, or the _____:

```
public boolean myMethod (...)
{
   ...
   if (...)
      return true;
   ...
   return n % 2 == 0;
}
```

_____

If its return type is a _____, the method returns a _____ (or _____).
Often the _____ is created in the method using _____. For example:

```
  public Fraction inverse ()
{
    if (num == 0)
       return null;
    return new Fraction (denom, num);
}
```

The returned object can also come from a _____ or from a _____.

Summary/Reflection:

QUESTIONS | **NOTES:**

Overloaded Methods

Methods _____ that have the _____ name but _____ numbers or types of parameters are called _____ methods.
Use overloaded methods when they perform _____ tasks:

```
public void move (int x, int y)   { ... }
public void move (double x, double y)   { ... }
public void move (Point p)   { ... }

public Fraction add (int n)   { ... }
public Fraction add (Fraction other)   { ... }
```

The compiler treats overloaded methods as _____.
The compiler knows which one to call based on the _____ and the _____ of the _____.

```
Circle circle =  new Circle(5);
circle.move (50, 100);
Point center =
          new Point(50, 100);
circle.move (center);
```

```
public class Circle
{
    public void move (int x, int y)
     { ... }

    public void move (Point p)
     { ... }
    ...
```

Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

The _____ alone is not sufficient for distinguishing between overloaded methods.

```
public class Circle
{
    public void move (int x, int y)
     { ... }

    public Point move (int x, int y)
     { ... }
    ...
```

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

Static Fields

A *static field* (a.k.a. *class field* or *class variable*) is _____ by _____ of the _____.
A non-static field (a.k.a. *instance field* or *instance variable*) belongs to an _____.

A static field can hold a _____ shared by all objects of the class:

```
public class RollingDie
{
   private static final double slowDown = 0.97;
   private static final double speedFactor = 0.04;
   ...
```

A static field can be used to collect statistics or totals for _____
(for example, total sales for all vending machines)

Static fields are stored with the _____, separately from instance variables that describe an individual object.

public static fields, usually _____, are referred to in other classes using "_____": ClassName.constName

```
double area = Math.PI * r * r;
setBackground(Color.BLUE);
c.add(btn, BorderLayout.NORTH);
System.out.println(area);
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Usually static fields are NOT initialized in _____ (they are initialized either in _____ or in _____).

If a class has only static fields, there is no point in creating objects of that class (_____).

_____ and _____ are examples of the above. They have no public constructors and cannot be instantiated.

Static Methods

Static methods can access and manipulate a _____.

Static methods _____ access non-static fields or call non-static methods of the class.

Static _____ are called using "dot notation": ClassName.statMethod(...)

```
double x = Math.random();
double y = Math.sqrt (x);
double avgGPA = Student.computeSchoolGPA ();
Vic.stackHasCD();
```

Summary/Reflection:

QUESTIONS | **NOTES:**

```
public class MyClass
{
    public static final int statConst;
    private static int statVar;

    private int instVar;
    ...
    public static int statMethod(...)
    {
        statVar = statConst;
        statMethod2(...);

        instVar = ...;
        instMethod(...);
    }
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Non-Static Methods

A _____ method is called for a particular object using "_____":

objName.instMethod(...);

Non-static methods can access _____ and call _____ of their class — both static and non-static.

Summary/Reflection:

Essential Question:

QUESTIONS	NOTES:

The String Class

Def. String – a _____ stored as one object.

Example

A	P		C	o	m	p	u	t	e	r
0	1	2	3	4	5	6	7	8	9	10

Strings are **_____**. That is once a String object is created, _____ of its methods can _____ it.

Strings are "Comparable"
java.lang.String _____ java.lang.Comparable

What does this mean??
Comparable is an **_____**, meaning that there are methods defined, but not _____ for the class.

interface java.lang.Comparable
int compareTo(Object other)
// _____ if this is less than other
// _____ if this is equal to other
// _____ if this is greater than other

Summary/Reflection:

QUESTIONS	**NOTES:**

How are Comparable objects used??

Ex. Write a method that returns the greatest of two String objects (in lexicographical order)

```
public String greatest (String s1, String s2)
{
```

Summary/Reflection:

QUESTIONS	**NOTES:**

Other String methods.

```
int length() // returns the # of characters in String

String substring(int from, int to)
// returns the substring beginning at from
// and ending at to-1

String substring(int from)
// returns substring(from, length())

int indexOf(String s)
// returns the index of the first occurrence of s;
// returns -1 if not found
```

See page 265 in your book for a list of other methods

Ex. Analyze the following code segment:

```
String s = "Help me, I am lost!!";

int a = s.length();
String b = s.substring(5, 14);
String c = s.substring(13);
int d = s.indexOf("me");
int e = s.indexOf("z");
```

Summary/Reflection:

Name: _____
Class: _____
Date: _____

QUESTIONS	**NOTES:**

String Constructors

String's no-args and copy constructors are _____.

```
String s1 = new String ();
```

```
String s2 = s1;
```

Not to be confused with an _____ string:

```
private String errorMsg;
```

errorMsg is

_____

Escape Sequences

Sequence	Meaning
\n	_____
\t	_____
\'	_____
\"	_____
\\	_____

```
System.out.print("\nDon\'t let me down\nDon\'t let me do
```

Summary/Reflection:

Essential Question:

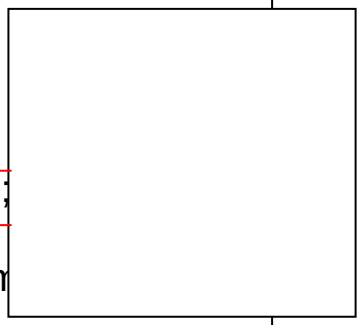
QUESTIONS	NOTES:

Methods — Concatenation

String result = ;
concatenates s1 and s2
String result = ;
the same as s1 + s2
result += ;
concatenates s3 to result
r**esult** += ;
converts **num** to **String** and concatenates it to result

Numbers to Strings

- Three ways to convert a number into a string:
  1. String s = "" + num;
  2. String s = Integer.toString (i);
     String s = Double.toString (d);
  3. String s = String.valueOf (num



Summary/Reflection:

QUESTIONS	**NOTES:**

Numbers from Strings

```
String s1 = "-123", s2 = "123.45";
int n = Integer.parseInt(s1);
double x = Double.parseDouble(s2);
```

These methods throw a _____ if s does not represent a _____
(integer, real number, respectively).

Summary/Reflection:

QUESTIONS	**NOTES:**
	String JavaBat Examples

Summary/Reflection: