How many iterations?

for (count = 1; count < 10; count++)
- Loop 9 times

for (count = 1; count <= 10; count++)
for (count = 0; count <= 10; count++)
for (count = 0; count < 10; count++)

for (count = 1; count < 10; count+=2)
More examples with String & Loop

• String – internally, a list of character
• Loop provides a good way to scan through the list

```java
String str = “COMP110”;
System.out.println(“Index\tChar”);
for (int i = 0; i<str.length(); i++) {
    System.out.println( i + “\t” + str.charAt(i) );
}
```

<table>
<thead>
<tr>
<th>Index</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>O</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>
More examples with String & Loop

• Find a character in String (both the string & character can come from user input or other external sources at run-time)

```java
String str = "COMP110";
char findme = 'P';

for (int i = 0; i<str.length(); i++) {
    if (str.charAt(i) == findme) {
        System.out.println(findme + " is found in " + str);
        break;
    }
}
```
More examples with String & Loop

String str = "COMP110";
char findme = 'P';
boolean found = false;
for (int i = 0; i < str.length(); i++) {
    if (str.charAt(i) == findme) {
        found = true;
        break;
    }
}
if (found) {
    System.out.println(findme + " is found in " + str);
} else {
    System.out.println(findme + " is NOT found in " + str);
}
More examples with String & Loop

• Count the occurrence of one character in a String

```java
String str = "COMP110";
char findme = '1';

int occurrence = 0;
for (int i = 0; i<str.length(); i++) {
    if (str.charAt(i) == findme) {
        occurrence ++;
    }
}
```
More examples with String & Loop

• Find whether a string is in another string
• Optional take-home question (CS students should work this out)

```java
String str = "A Longer String Here";
String findme = "ger";

for(int i = 0; i<str.length(); i++) {
    .... Fill out the code here
    .... Inner loop is needed
}
```
Classes and Objects

Review of concepts

• Java programs (and programs in other object-oriented programming languages) consist of objects of various class types

• Objects can represent objects in the real world
  – Automobiles, houses, employee records

• Or abstract concepts
  – Colors, shapes, words
Object Oriented Programming (OOP)

- **Object**: Attributes + Methods
- **Class**: the blueprint of objects of the same type

### Diagram

- **Superclass**: Person
  - name, contact
- **Subclass 1**: Student
  - student ID, program, year
- **Subclass 2**: Teacher
  - employee ID, department, rank

#### Objects

- **S1**
  - name="Alan",
  - contact="919-......",
  - program = biostat,
  - year = 1st

- **S2**
  - name="Anna",
  - contact="919-......",
  - program = CS,
  - year = 1st

- **T1**
  - name="Yi",
  - contact="919-......",
  - dept = CS,
  - rank = no rank

- **T2**
  - name="Yun",
  - contact="919-......",
  - program = biostat,
  - rank = ast prof
OOP in Practice

• Import class if necessary:
  `import java.util.*;`

• Create object:
  `Class_Type variable_name = new Class_Type( ... );`
  e.g.:
  `Scanner s = new Scanner(System.in);
  Polygon treeTop = new Polygon();`

• Access object members (attribute or method):
  `int inputNumber = s.nextInt();`
  `treeTop.setColor( Color.green );`
Class

• Today, we will talk about how to create our own classes

• A class is the definition of a kind of object
  – A blueprint for constructing specific objects

Class Name: Automobile

Data:
  amount of fuel
  speed
  license plate

Methods (actions):
  accelerate:
    How: Press on gas pedal.
  decelerate:
    How: Press on brake pedal.
Objects, Instantiation

Object Name: patsCar
amount of fuel: 10 gallons
speed: 55 miles per hour
license plate: “135 XJK”

Object Name: suesCar
amount of fuel: 14 gallons
speed: 0 miles per hour
license plate: “SUES CAR”

Object Name: ronsCar
amount of fuel: 2 gallons
speed: 75 miles per hour
license plate: “351 WLF”

Instantiations, or instances, of the class Automobile
### UML (Universal Modeling Language)

**Class name**: Automobile

**Data**:
- fuel: `double`
- speed: `double`
- license: `String`

**Methods (actions)**:
- `+ accelerate(double pedalPressure): void`
- `+ decelerate(double pedalPressure): void`
Objects

• **Important**: classes do not have data; individual objects have data

• Classes specify what kind of data objects have
Class files

• Each Java class definition goes in its own, SEPARATE .java file

• ClassName ➔ save the file as ClassName.java

• Student.java includes the class Student
Class files and separate compilation

• What happens when you compile a .java file?
  – .java file gets compiled into a .class file
    • Contains Java bytecode
    • Same filename except for .class instead of .java

• You can compile a Java class before you have a program that uses it

• Don’t worry about the compilation in this course as Eclipse does it automatically

• For CS students, I may hold an extra class to talk about how to compile / run programs in command line window.
class Student

<table>
<thead>
<tr>
<th>Class Name: Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Name</td>
</tr>
<tr>
<td>- Year</td>
</tr>
<tr>
<td>- GPA</td>
</tr>
<tr>
<td>- Major</td>
</tr>
<tr>
<td>- Credits</td>
</tr>
<tr>
<td>- GPA sum</td>
</tr>
</tbody>
</table>

+ getName
+ getMajor
+ printData
+ increaseYear
  How: increase year by 1
+ calcGpa
  How: average grades
# class Student

<table>
<thead>
<tr>
<th>Class Name:</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>- name:</td>
<td>String</td>
</tr>
<tr>
<td>- year:</td>
<td>int</td>
</tr>
<tr>
<td>- gpa:</td>
<td>double</td>
</tr>
<tr>
<td>- major:</td>
<td>String</td>
</tr>
<tr>
<td>- credits:</td>
<td>int</td>
</tr>
<tr>
<td>- gpaSum:</td>
<td>double</td>
</tr>
</tbody>
</table>

+ getName(): String
+ getMajor(): String
+ printData(): void
+ increaseYear(): void
+ calcGpa(double grade): void
public class Student {
    public String name;
    public int classYear;
    public double GPA;
    public String major;
    // ...

    public String getMajor() {
        return major;
    }

    public void increaseYear() {
        classYear++;
    }
}
Creating an object

Create an object *jack* of class *Student*

Student jack = new Student();

Scanner keyboard = new Scanner(System.in);

Create an object *keyboard* of class *Scanner*
Instance variables

• Data defined in the class are called *instance variables*

```java
public String name;
public int classYear;
public double GPA;
public String major;
```

**public**: no restrictions on how these instance variables are used (more details later – **public** is actually a bad idea here)

**type**: `int`, `double`, `String`...
Public Class Student {
    public String name;
    public Student Friend;
}

Using the Student class:

Student s1 = new Student();
s1.name = "Alan";
...
Student s2 = new Student();
s2.name = "Anna";
...
s1.Friend = s2;
Next Class

• Continue on Class
• Define methods