# First version of a Student Class

- For each student, we want to store their ID number, their midterm score, their exam score, and whether or not the student is t

| Student |
| --- |
| id : int<br>midterm : double<br>exam : double<br>forCredit : boolean |
| (no methods yet!) |

1

# Calculation of the Final Mark

- Write a Java method **`getFinalMark( )`** for our **`Student`** class that returns a **`double`** with a student's final mark, where:
  - The final mark is 55% of the exam, plus 20% of the midterm, plus 25% of the average of 5 assignments.

```
public double getFinalMark()
{
  double assignAvg;
  double assignSum = 0.0;
  int numAssigns = 5;              // constant
  int index;
  for ( index = 0; index < numAssigns; index = index + 1 )
  {
    assignSum = assignSum + this.getAssignment( index );
  }
  assignAvg = assignSum / (double) numAssigns  // mixed types here
  double finalMark = 0.55 * this.getExam( ) + 0.2 * this.getMidterm( )
              + 0.25 * assignAvg;
  return finalMark;
}
```

2

# Course information

- Now that we have a class that stores information about one Student, how can we use this create a class Course that stores information about all students?

| Course |
| --- |
| students : Student [ ] |
| (no methods yet!) |

3

# Designing a Fraction class

- What information do we need to store in a Fraction?
  - numerator
  - denominator
- What operations do we need?
  - [Aside from creating fractions, the only mathematical operation we will implement is addition of two fractions]
  - constructor(s)
    - When constructing a Fraction, we may need some additional methods
  - display( )
  - addTo( )

4

# Designing a Fraction class

| Fraction |
| --- |
| – numerator : int<br>– denominator : int |
| + Fraction( a : int )<br>+ Fraction( n : int, d : int )<br>+ display( )<br>+ addTo( operand : Fraction ) : Fraction<br>– gcd( a : int, b : int )<br>– simplify( ) |

5

# Putting the Fraction in Standard Form

- To make sure that each **`Fraction`** instance will be in lowest terms, a method **`simplify`** will be used.
- Assume that you have a method **`gcd(a,b)`** that will return the greatest common divisor of two integers.
- Write a Java method to put a fraction into standard form.

```
private void simplify( )
{
   int f;
   f = gcd(numerator, denominator);
   if (f != 0)
   {
      numerator = numerator / f;
      denominator = denominator / f;
   }
   else
   {
      ; // do nothing
   }
   if (denominator < 0)
   {
     numerator = -numerator;
     denominator = -denominator;
   }
   else
   {
     ; // do nothing
   }
}
```

6

# Algorithm for GCD

- A recursive GCD algorithm for gcd(a,b):
  - If a mod b is 0, gcd(a, b) is b
    - a mod b is the remainder when a is divided by b
  - Otherwise, gcd(a,b) is gcd(b, a mod b)
- Question: will this algorithm always reach the base case?
  - Note that a mod b is at most b – 1.
- Write a recursive Java method to calculate gcd(a,b)

```
private static int gcd (int a, int b)   // a class method
{
   int result;
   int remainder;
   remainder = a % b;
   if ( remainder == 0 )
   {
      result = b;
   }
   else
   {
      result = gcd( b, remainder );
   }
   return result;
}
```

7

# Fraction Constructors

- Write constructors for a **`Fraction`** that:
  - take 2 integers: the numerator and the denominator
  - takes 1 integer, representing an integer that is to be converted to a Fraction

```
public Fraction(int n, int d)
{
   numerator = n;
   denominator = d;
   simplify( );
}
public Fraction(int a)
{
   numerator = a;
   denominator = 1;
}
```

8

# Displaying Fractions

- Write a Java method to display a **Fraction**.
- Sample usage:
  - `Fraction f1 = new Fraction ( 6, -9 );`
  - `f1.display( );`
- Result: **-2/3**

```
public void display( )
{
   if (denominator != 1)
   {
      System.out.print(numerator + " / " + denominator);
   }
   else
   {
      System.out.print(numerator);
   }
}
```

# Adding Fractions

- Write a Java method that will add two Fractions.
  - Sample usage:

```
Fraction f1 = new Fraction( 1, 2 );
Fraction f2 = new Fraction( 1, 3 );
Fraction sum = f1.addTo( f2 );
sum.display( );
```

  - Result: **5/6**

```
public Fraction addTo(Fraction operand)
{
   int n = numerator * operand.denominator
           + denominator * operand.numerator;
   int d = denominator * operand.denominator;
   return new Fraction(n, d);
}
```