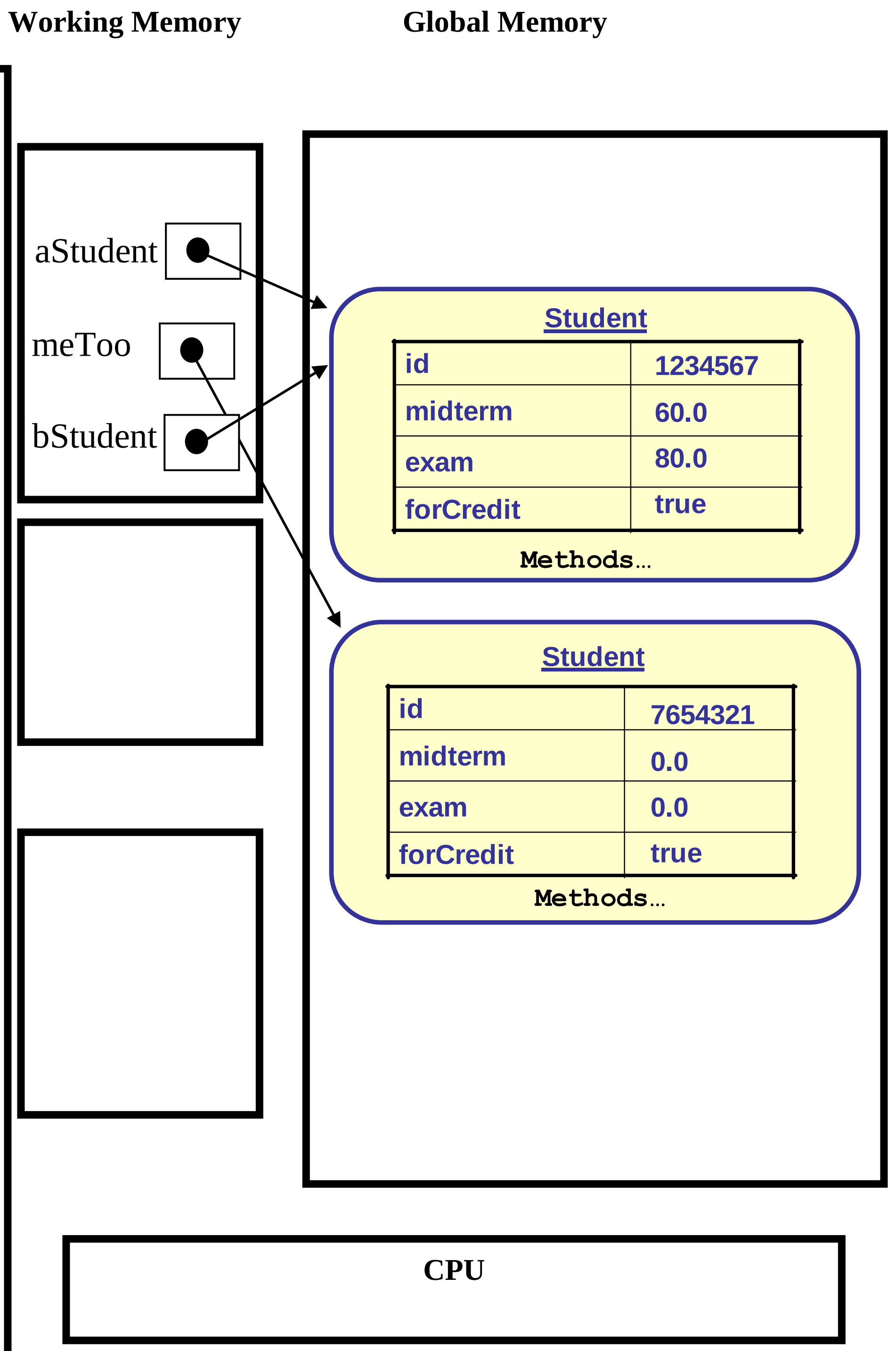
1. Section 11 Exercises

Program Memory Exercise 11-1 - Using Constructors

```
public class Section11
{
   public static void main(String [] args)
   {

     Student aStudent; // reference variable
     Student meToo;    // another reference variable
     Student bStudent; // a third reference variable
              •
              •
     aStudent = new Student(1234567,60.0,80.0,true);
     meToo = new Student(7654321,true);
     bStudent = aStudent;
              •
              •
              •
   }
}

class Student
{
   // ... fields would be defined here ...
   public Student(int theId, double theMidterm,
                  double theExam, boolean isForCredit)
   {
      this.id = theId;
      this.midterm = theMidterm;
      this.exam = theExam;
      this.forCredit = isForCredit;
   }
   public Student(int theID, boolean isForCredit )
   {
      this.id = theID;
      this.midterm = 0.0;     // a "safe" value
      this.exam = 0.0;        // a "safe" value
      this.forCredit = isForCredit;
   }
```



Program Memory **Exercise 11-2 – Calculation of the Final Mark** Working Memory Global Memory

```
public class Student
{
   // Attributes
   private int id;
   private double midterm;
   private double exam;
   private boolean forCredit;
   private double [] assignments;
   // Methods
   public double calcAssignAvg()
   {
      double sumAssigns;
      int numAssigns = 5;// constant
      int index;
      sumAssigns = 0.0
      for ( index = 0; index < numAssigns;
            index = index + 1 )
      {
         sumAssigns = sumAssigns +
                   this.getAssignment(index);
      }
      avgAssigns = sumAssigns /(double)numAssigns
                             // mixed types here
      return avgAssigns;
   }

   public double getFinalMark()
   {
      double assignAvg;
      int index;
      assignAvg = calcAssignAvg();
      double finalMark = 0.55 * this.getExam( )
                     + 0.2 * this.getMidterm( )
                     + 0.25 * assignAvg;
      return finalMark;
   }
} // end of class Student
```

CPU

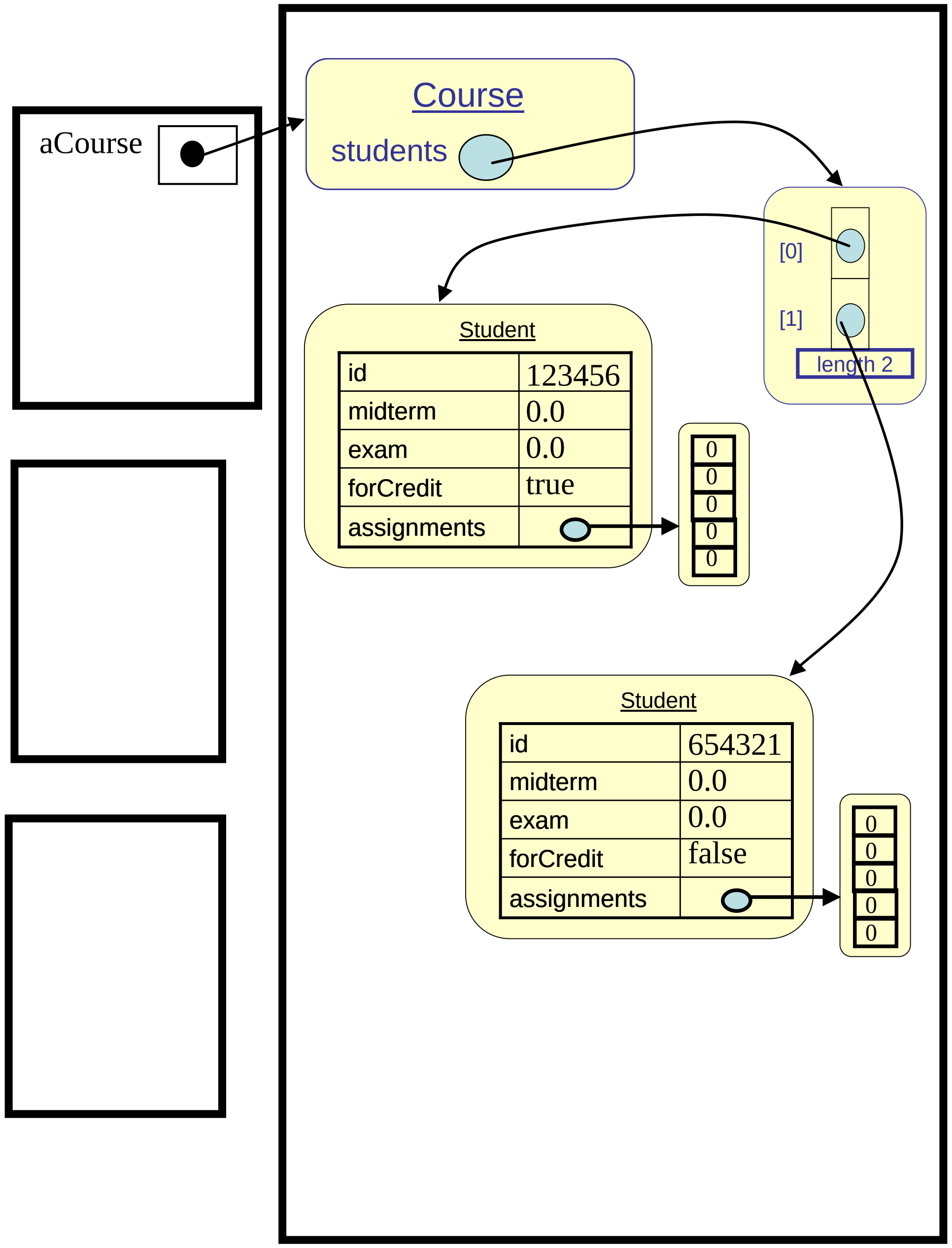
Program Memory **Exercise 11-3 - Arrays of Objects** Working Memory Global Memory

```
public class Section11
{
   public static void main(String [] args)
   {
     Course aCourse;

     aCourse = new Course();

     aCourse.addStudent(123456,true);
     aCourse.addStudent(654321,false);
   }
}
```

Course

– code : String
– title : String
– students: student [ ]
…

+ Course(code: String, title: String)
+ addStudent(theID : int, isForCredit : boolean)



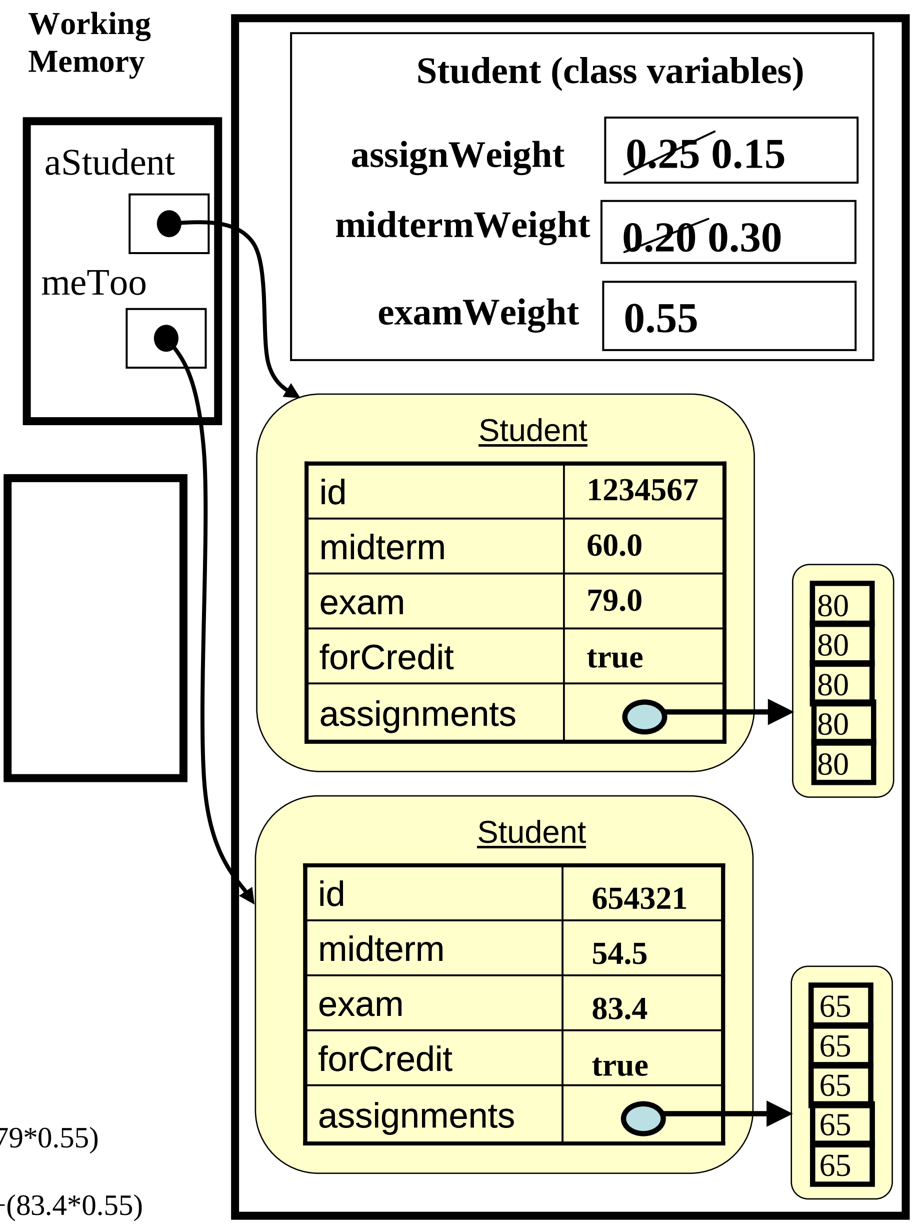
CPU

Program Memory

Exercise 11-4 - Using Class Variable and Methods

Global Memory

```
public class Section11
{
  public static void main(String [] args)
  {
    int anum;
    Student aStudent; // reference variable
    Student meToo;    // another reference variable
    aStudent = new Student(1234567,60.0,79.0,true);
    meToo = new Student(7654321,54.5, 83.4, true);
    for(anum=0 ; anum<5 , num=i+1)
    {
      aStudent.setAssignment(anum, 80.0);
      meToo.setAssignment(anum, 65.0);
    }
    System.out.println(“The mark for student “ +
                       aStudent.getId()+ “ is “+
                       aStudent.getFinalMark();
    Student.SetMidWeight(0.30);
    Student.SetAssignWeight(0.15);
    System.out.println(“The mark for student “ +
                      meToo.getId()+ “ is “+
                      meToo.getFinalMark();
    System.out.println(“The mark for student “ +
                      aStudent.getId()+ “ is “+
                      aStudent.getFinalMark();
  }
}
```



Terminal Window

```
The mark for student 1234567 is 75.45
The mark for student 7654321 is 71.97
The mark for student 1234567 is 73.45
```

(80*0.25)+(60*0.20)+(79*0.55)

(65*0.15)+(54.5*0.30)+(83.4*0.55)

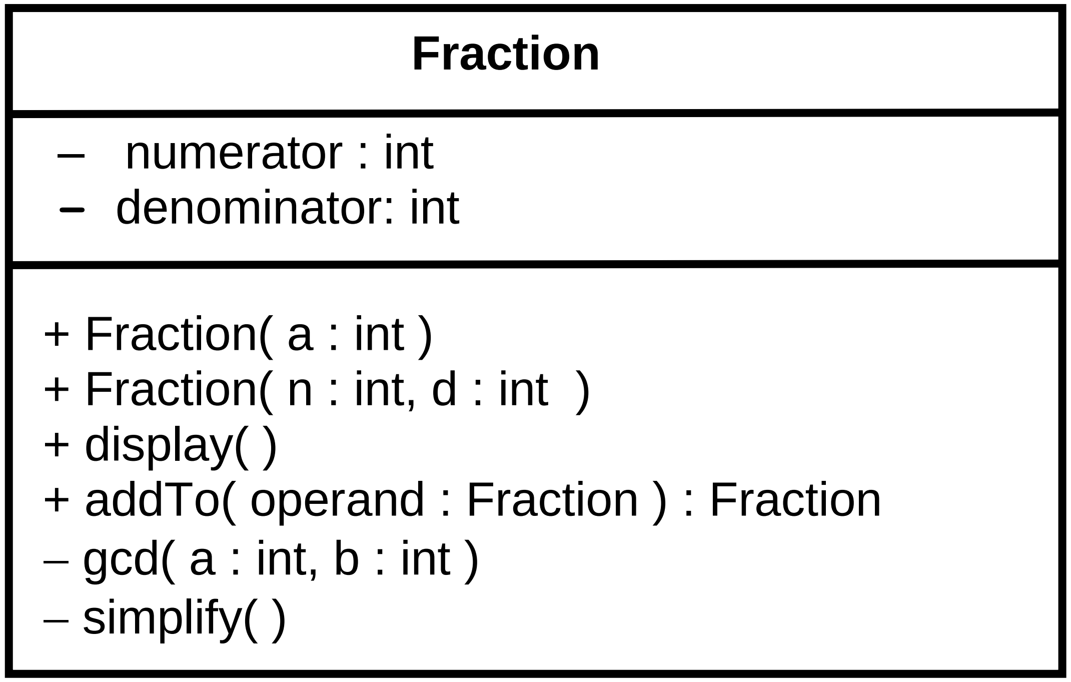
(80*0.15)+(60*0.30)+(79*0.55)

CPU

Exercise 11-5 – 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]
  - **Creation (with numerator, with numerator and denominator)**
  - **Addition**
  - **GCD (greatest common denominator) – support method**
  - **Reduction – support method**



Fraction

– numerator : int
– denominator: int

+ Fraction(a : int)
+ Fraction(n : int, d : int)
+ display( )
+ addTo(operand : Fraction) : Fraction
– gcd(a : int, b : int)
– simplify( )

Program Memory

Exercise 11-6 – Simplify Fraction to Standard Form

Global Memory

```
 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 */;
  }
}
```

Working Memory

CPU

Program Memory

Exercise 11-7 – Method for GCD

Working Memory

Global Memory

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

CPU

Program Memory **Exercise 11-8 – Fraction Constructors** Working Memory Global Memory

```
public class Fraction
{
  private int numerator;
  private int denominator;

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

  public Fraction(int a)
  {
   numerator = a;
   denominator = 1;
 }
```

Exercise 11-9 Displaying Fractions

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

CPU

Program Memory

Exercise 11-10 - Adding Fractions

Working Memory **Global Memory**

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

Exercise 11-11: Adding an Integer to a Fraction

```
public Fraction addTo(int integer)
{
     return this.addTo(new Fraction(integer));
    // this is optional here
}


Public String toString()
{
   String result;
   if(denominator != 1)
   {
      result = numerator +
                  " / " + denominator);
   }
   else
   { result = String.valueOf(numerator); }
   return result;
}
```

CPU

Program Memory

Exercise- Using Fraction

Working Memory

Global Memory

```
class FractionsDriverClass
{
    public static void main (String[] args)
     {
      Fraction f1, f2, f3;

      f1 = new Fraction(6, 9);
      f2 = new Fraction(24, 46);
      f3 = f1.addTo(f2);

      f1.display();
      f2.display();
      f3.display();

      System.out.println(f1);
      System.out.println(f2);

     }
}
```

Console Display

```
2 / 3
12 / 23
82 / 69
2 / 3
12 / 23
```

CPU