

Class 6: Lab Session on Classes and Objects

1.00/1.001 - Introduction to
Computation and Problem Solving

Using Methods

- **Methods are invoked using the dot (.) operator**
 - Method always ends with parentheses

```
Bi g l n t e g e r  a =  n e w  B i g l n t e g e r ("1000000000000");
Bi g l n t e g e r  z =  n e w  B i g l n t e g e r ("23");
Bi g l n t e g e r  c =  a . a d d (z);           // c = a + z
I f  ( z . i s P r o b a b l e P r i m e (15))    // i s z p r i m e ?
    S y s t e m . o u t . p r i n t l n ("z i s p r o b a b l y p r i m e");
```

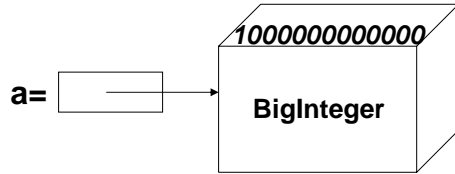
- **Public data fields are also invoked with the dot operator.**

- No parentheses after field name

```
i n t  j =  a . s o m e P u b l i c F i e l d ;           // E x a m p l e  o n l y
```

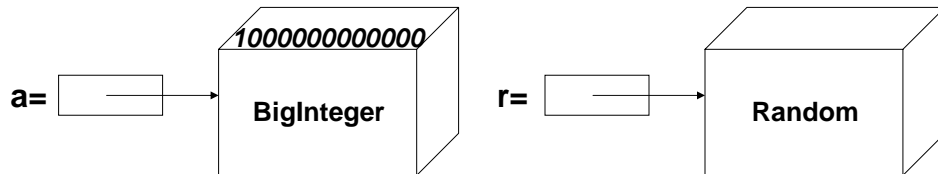
Objects and Names

```
BigInteger a= new BigInteger("1000000000000");
```



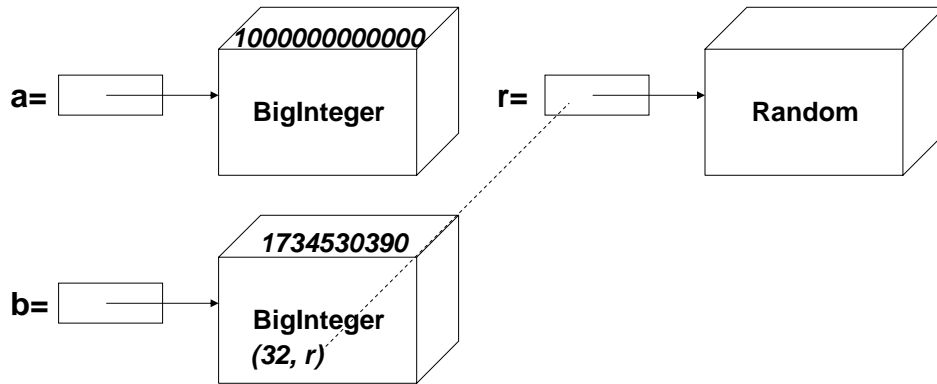
Objects and Names

```
BigInteger a= new BigInteger("1000000000000");  
Random r= new Random();
```



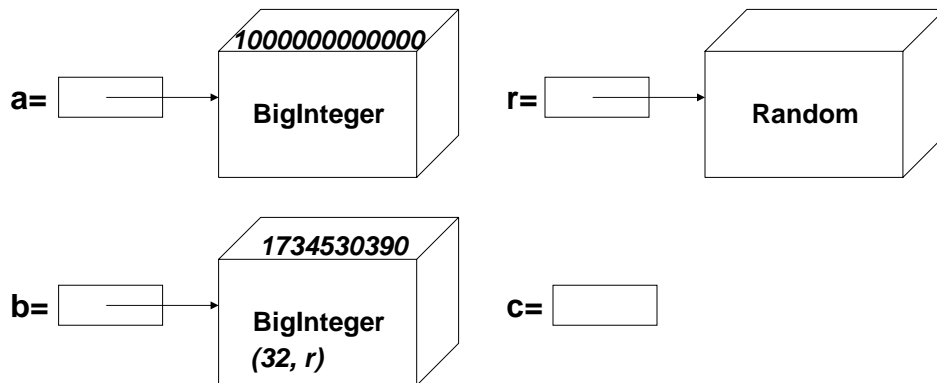
Objects and Names

```
BigInteger a= new BigInteger("1000000000000");  
Random r= new Random();  
BigInteger b= new BigInteger(32, r);
```



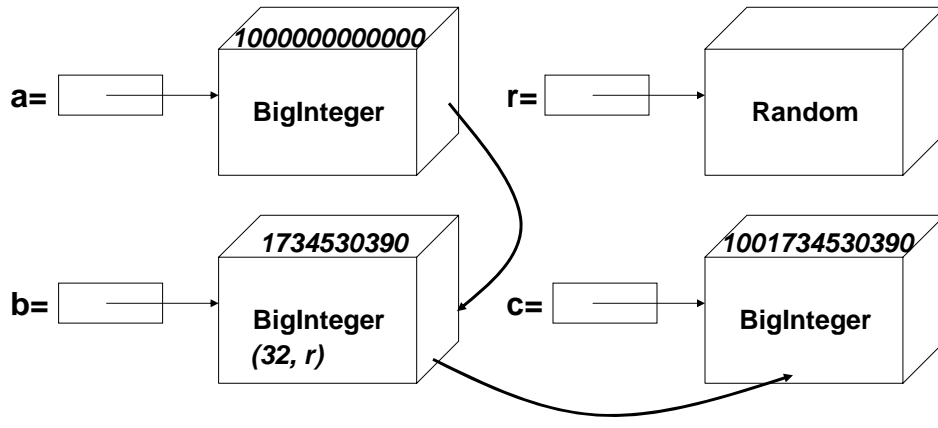
Objects and Names

```
BigInteger a= new BigInteger("1000000000000");  
Random r= new Random();  
BigInteger b= new BigInteger(32, r);  
BigInteger c;
```



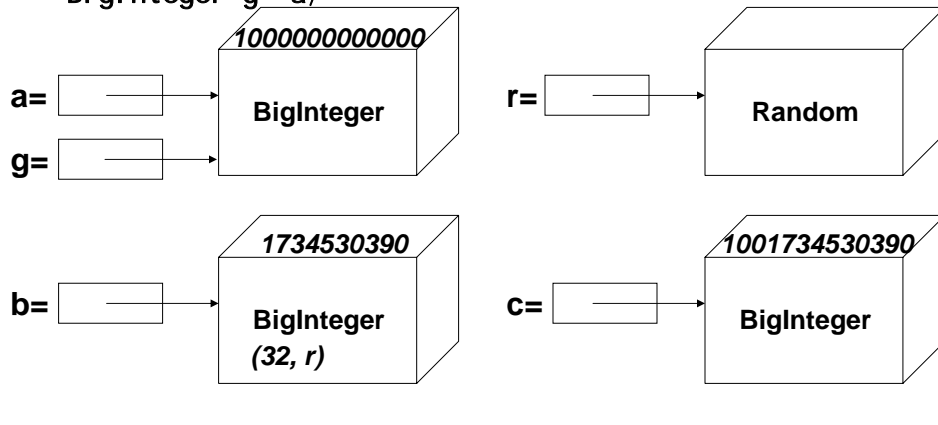
Objects and Names

```
BigInteger a= new BigInteger("1000000000000");  
Random r= new Random();  
BigInteger b= new BigInteger(32, r);  
BigInteger c;  
c= b.add(a);
```



Objects and Names

```
BigInteger a= new BigInteger("1000000000000");  
Random r= new Random();  
BigInteger b= new BigInteger(32, r);  
BigInteger c;  
c= b.add(a);  
BigInteger g= a;
```



Using the BigInteger Class

```
import java.math.*;           // For BigInteger
import java.util.*;         // For Random
public class BigIntTest {
    public static void main(String[] args) {
        BigInteger a= new BigInteger("1000000000000");
        Random r= new Random(); // Random nbr generator
        BigInteger b= new BigInteger(32, r); // Random
        BigInteger c;
        c= b.add(a); // c= b+a
        BigInteger g= a;
        BigInteger d= new BigInteger(32, 10, r); // Prime
        BigInteger e;
        e= c.divide(d); // e= c/d
        if (d.isProbablePrime(10))
            System.out.println("d is probably prime");
        else
            System.out.println("d is probably not prime");
        BigInteger f= d.multiply(e); // f= d*e
    }
}
```

Exercise 1: Existing Class

- Use the **BigDecimal** class (floating point numbers) to:
 - Construct **BigDecimal a= 13 x 10³⁰**
 - Construct **BigDecimal b** randomly
 - Hint: Construct a random **BigInteger**, then use the appropriate **BigDecimal** constructor. See Javadoc
 - Compute **BigDecimal c= a + b**
 - Print out **a, b, c, d** after computing each one

Exercise 1: Existing Class (2)

- **Write the program in stages:**
 - **Construct a, print it. Compile and debug**
 - **Don't count the zeros!**
 - **After constructing b, print it. Compile and debug**
 - **Do the addition and division. Compile and debug**

Exercise 2: Writing A Class

- **In homeworks, you will be writing your own classes**
 - **You've already seen classes in all our examples, but they're not typical**
 - **They just have a single method, main()**
 - **Most classes don't have a main() method**
- **To build a program, you'll write several classes, one of which has a main() method**

Point Class

```
public class SimplePoint {
    private double x, y;           // Data members
    public SimplePoint() {        // Constructor
        x= 0.0;
        y= 0.0; }
    // Methods
    public double getX() { return x;}
    public double getY() { return y;}
    public void setX(double xval) { x= xval;}
    public void setY(double yval) { y= yval;}
    public void move(double del taX, double del taY) {
        x += del taX;
        y += del taY; }
} // End of class SimplePoint

// This isn't a program because it doesn't have main()
// but it can be used by classes with a main()
```

Point Class, main()

```
public class SimplePoint1 {
    public static void main(String[] args) {
        SimplePoint a= new SimplePoint();
        SimplePoint b= new SimplePoint();
        double xa= a.getX();
        double ya= a.getY();
        System.out.println("a= (" + xa + " , " + ya + ")");
        a.move(-9.0, 7.5);
        System.out.println("a= (" + a.getX() +
            " , " + a.getY() + ")");
    }
}
```

Exercise 2

- Write a different SimplePoint class that stores the point's position in polar coordinates instead of Cartesian coordinates
 - Implement the same public methods as the previous SimplePoint class
 - Use r and θ as the private data fields
 - Recall that:
 - $x = r \cos(\theta)$
 - $y = r \sin(\theta)$
 - $r = \sqrt{x^2 + y^2}$
 - $\theta = \tan^{-1}(y/x)$
 - Use the Java Math class (capital M)
 - Use `Math.atan2()` for the arctan function
- Use the same `main()` as before

Why Do This?

- By building a class with public methods but private data, you only commit to an interface, not an implementation
 - If you need to change implementation, you can do so without breaking any code that depends on it, as long as the interface (set of methods) stays the same
 - Changing coordinate systems, computational methods, etc., is quite common, as in this example. This allows flexibility as software grows and changes

Exercise 3-Using Point Class

- Create a new class called Rectangle that has two Point objects in it, one for the upper left corner and one for the lower right corner
- Write a setUpperLeft() and setLowerRight() method that has double values as arguments
- Write a getArea() method that returns the area of the rectangle as a double

Point Class, Polar Coordinates

```
class SimplePoint {
    private double r, theta;    // Data members
    public SimplePoint() {     // Constructor
        r= 0.0;
        theta= 0.0; }
    // Methods (trig functions use radians)
    public double getX() { return r* Math.cos(theta); }
    public double getY() { return r* Math.sin(theta); }
    public void setX(double xval) {
        double yval = r*Math.sin(theta);
        r= Math.sqrt(xval*xval + yval*yval);
        theta= Math.atan2(yval, xval); }
}
```

Point Class, Polar, p.2

```
public void setY(double yval) {
    double xval = r*Math.cos(theta);
    r= Math.sqrt(xval*xval + yval*yval);
    theta= Math.atan2(yval, xval); }
public void move(double del taX, double del taY) {
    double xval = r*Math.cos(theta);
    double yval = r*Math.sin(theta);
    xval += del taX;
    yval += del taY;
    r= Math.sqrt(xval*xval + yval*yval);
    theta= Math.atan2(yval, xval);
}
}

// Can be invoked from same main() as before and
// produces the same results (other than rounding errors)
```

```
public class Rectangle {
    private SimplePoint upperLeft, lowerRight;
    public Rectangle() { // Constructor
        upperLeft = new SimplePoint();
        lowerRight = new SimplePoint(); }
    public SimplePoint getUpperLeft() {return upperLeft;}
    public SimplePoint getLowerRight() {return lowerRight;}
    public void setUpperLeft(double xval, double yval)
    { upperLeft.setX(xval);
      upperLeft.setY(yval); }
    public void setLowerRight(double xval, double yval)
    { lowerRight.setX(xval);
      lowerRight.setY(yval); }
    public double getArea() {
        double area = (upperLeft.getX()-lowerRight.getX())
            * (upperLeft.getY()-lowerRight.getY());
        return Math.abs(area);
    } } // End of class Rectangle
```

Point Class, Polar Coordinates

```
class SimplePoint {
    private double r, theta;    // Data members
    public SimplePoint() {     // Constructor
        r= 0.0;
        theta= 0.0; }
    // Methods (trig functions use radians)
    public double getX() { return r* Math.cos(theta);}
    public double getY() { return r* Math.sin(theta);}
    public void setX(double xval) {
        double yval = r*Math.sin(theta);
        r= Math.sqrt(xval*xval + yval*yval);
        theta= Math.atan2(yval, xval); }
}
```

```
import java.util.*;
import java.math.*;

public class BigDecTest {
    public static void main(String[] args) {
        BigDecimal a= new BigDecimal ("13E500");
        System.out.println("a: " + a);
        Random r= new Random();
        BigInteger i= new BigInteger(64, r);
        System.out.println("i: " + i);
        BigDecimal b= new BigDecimal (i);
        System.out.println("b: " + b);
        BigDecimal c= a.add(b);
        System.out.println("c: " + c);
        BigDecimal d= c.divide(a, BigDecimal.ROUND_UP);
        System.out.println("d: " + d);
        System.exit(0);
    }
}
```

Point Class, Polar, p.2

```
public void setY(double yval) {
    double xval = r*Math.cos(theta);
    r= Math.sqrt(xval*xval + yval*yval);
    theta= Math.atan2(yval, xval); }
public void move(double del taX, double del taY) {
    double xval = r*Math.cos(theta);
    double yval = r*Math.sin(theta);
    xval += del taX;
    yval += del taY;
    r= Math.sqrt(xval*xval + yval*yval);
    theta= Math.atan2(yval, xval);
}
}

// Can be invoked from same main() as before and
// produces the same results (other than rounding errors)
```