3.2 Creating Data Types
Data Types

**Data type.** Set of values and operations on those values.

**Basic types.**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Set of Values</th>
<th>Some Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true, false</td>
<td>not, and, or, xor</td>
</tr>
<tr>
<td>int</td>
<td>(-2^{31} \text{ to } 2^{31} - 1)</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>String</td>
<td>sequence of Unicode characters</td>
<td>concatenate, compare</td>
</tr>
</tbody>
</table>

**Last time.** Write programs that *use* data types.
**Today.** Write programs to *create* our own data types.
Defining Data Types in Java

To define a data type, specify:

- Set of values.
- Operations defined on those values.

Java class. Defines a data type by specifying:

- Instance variables. (set of values)
- Methods. (operations defined on those values)
- Constructors. (create and initialize new objects)
Turtle Graphics
Turtle Graphics

Goal. Create a data type to manipulate a turtle moving in the plane.

Set of values. Location and orientation of turtle.

API.

```java
public class Turtle {
    Turtle(double x0, double y0, double a0)
        create a new turtle at \((x_0, y_0)\) facing \(a_0\) degrees counterclockwise from the \(x\)-axis
    void turnLeft(double delta)
        rotate \(\delta\) degrees counterclockwise
    void goForward(double step)
        move distance \(\text{step}\), drawing a line

    // draw a square
    Turtle turtle = new Turtle(0.0, 0.0, 0.0);
    turtle.goForward(1.0);
    turtle.turnLeft(90.0);
    turtle.goForward(1.0);
    turtle.turnLeft(90.0);
    turtle.goForward(1.0);
    turtle.turnLeft(90.0);
    turtle.goForward(1.0);
    turtle.turnLeft(90.0);
}```
public class Turtle {
    private double x, y; // turtle is at (x, y)
    private double angle; // facing this direction

    public Turtle(double x0, double y0, double a0) {
        x = x0;
        y = y0;
        angle = a0;
    }

    public void turnLeft(double delta) {
        angle += delta;
    }

    public void goForward(double d) {
        double oldx = x;
        double oldy = y;
        x += d * Math.cos(Math.toRadians(angle));
        y += d * Math.sin(Math.toRadians(angle));
        StdDraw.line(oldx, oldy, x, y);
    }
}
public class Ngon {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double angle = 360.0 / N;
        double step = Math.sin(Math.toRadians(angle/2.0));
        Turtle turtle = new Turtle(0.5, 0, angle/2.0);
        for (int i = 0; i < N; i++) {
            turtle.goForward(step);
            turtle.turnLeft(angle);
        }
    }
}
public class Spiral {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double decay = Double.parseDouble(args[1]);
        double angle = 360.0 / N;
        double step = Math.sin(Math.toRadians(angle/2.0));
        Turtle turtle = new Turtle(0.5, 0, angle/2.0);
        for (int i = 0; i < 10 * N; i++) {
            step /= decay;
            turtle.goForward(step);
            turtle.turnLeft(angle);
        }
    }
}
Spira Mirabilis in Nature
Complex Numbers
Complex Number Data Type

**Goal.** Create a data type to manipulate complex numbers.

**Set of values.** Two real numbers: real and imaginary parts.

**API.**

```java
public class Complex {
    Complex(double real, double imag)
    Complex plus(Complex b)  // sum of this number and b
    Complex times(Complex b) // product of this number and b
    double abs()             // magnitude
    String toString()        // string representation
}
```

\[ a = 3 + 4i, \ b = -2 + 3i \]
\[ a + b = 1 + 7i \]
\[ a \times b = -18 + i \]
\[ |a| = 5 \]
Applications of Complex Numbers

Relevance. A quintessential mathematical abstraction.

Applications.
- Fractals.
- Impedance in RLC circuits.
- Signal processing and Fourier analysis.
- Control theory and Laplace transforms.
- Quantum mechanics and Hilbert spaces.
- ...
Client program. Uses data type operations to calculate something.

```java
public static void main(String[] args) {
    Complex a = new Complex(3.0, 4.0);
    Complex b = new Complex(-2.0, 3.0);
    Complex c = a.times(b);
    StdOut.println("a = " + a);
    StdOut.println("b = " + b);
    StdOut.println("c = " + c);
}
```

```
% java TestClient
a = 3.0 + 4.0i
b = -2.0 + 3.0i
c = -18.0 + 1.0i
```

Remark. Can't write \( c = a * b \) since no operator overloading in Java.
Complex Number Data Type: Implementation

```java
public class Complex {
    private final double re;
    private final double im;  // instance variables

    public Complex(double real, double imag) {
        re = real;
        im = imag;
    }  // constructor

    public String toString() { return re + " + " + im + "i"; }
    public double abs() { return Math.sqrt(re*re + im*im); }

    public Complex plus(Complex b) {  // creates a Complex object, and returns a reference to it
        double real = re + b.re;
        double imag = im + b.im;
        return new Complex(real, imag);
    }

    public Complex times(Complex b) {  // refers to b's instance variable
        double real = re * b.re - im * b.im;
        double imag = re * b.im + im * b.re;
        return new Complex(real, imag);
    }
}
```
Mandelbrot set. A set of complex numbers.

Plot. Plot \((x, y)\) black if \(z = x + y \, i\) is in the set, and white otherwise.

- No simple formula describes which complex numbers are in set.
- Instead, describe using an algorithm.
Mandelbrot set. Is complex number $z_0$ in the set?

- Iterate $z_{t+1} = (z_t)^2 + z_0$.
- If $|z_t|$ diverges to infinity, then $z_0$ is not in set; otherwise $z_0$ is in set.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$Z_t$</th>
<th>$t$</th>
<th>$Z_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1/2 + 0i</td>
<td>0</td>
<td>1 + i</td>
</tr>
<tr>
<td>1</td>
<td>-1/4 + 0i</td>
<td>1</td>
<td>1 + 3i</td>
</tr>
<tr>
<td>2</td>
<td>-7/16 + 0i</td>
<td>2</td>
<td>-7 + 7i</td>
</tr>
<tr>
<td>3</td>
<td>-79/256 + 0i</td>
<td>3</td>
<td>1 - 97i</td>
</tr>
<tr>
<td>4</td>
<td>-26527/65536 + 0i</td>
<td>4</td>
<td>-9407 - 193i</td>
</tr>
<tr>
<td>5</td>
<td>-1443801919/4294967296 + 0i</td>
<td>5</td>
<td>88454401 + 3631103i</td>
</tr>
</tbody>
</table>

$z = -1/2$ is in Mandelbrot set

$z = 1 + i$ not in Mandelbrot set
Plotting the Mandelbrot Set

Practical issues.

- Cannot plot infinitely many points.
- Cannot iterate infinitely many times.

Approximate solution.

- Sample from an $N$-by-$N$ grid of points in the plane.
- Fact: if $|z_t| > 2$ for any $t$, then $z$ not in Mandelbrot set.
- Pseudo-fact: if $|z_{255}| \leq 2$ then $z$ "likely" in Mandelbrot set.
Mandelbrot function with complex numbers.

- Is $z_0$ in the Mandelbrot set?
- Returns white (definitely no) or black (probably yes).

```java
public static Color mand(Complex z0) {
    Complex z = z0;
    for (int t = 0; t < 255; t++) {
        if (z.abs() > 2.0) return StdDraw.WHITE;
        z = z.times(z);
        z = z.plus(z0);
    }
    return StdDraw.BLACK;
}
```

More dramatic picture: replace `StdDraw.WHITE` with grayscale or color. new Color(255-t, 255-t, 255-t)
Complex Number Data Type: Another Client

Plot the Mandelbrot set in gray scale.

```java
public static void main(String[] args) {
    double xc = Double.parseDouble(args[0]);
    double yc = Double.parseDouble(args[1]);
    double size = Double.parseDouble(args[2]);
    int N = 512;
    Picture pic = new Picture(N, N);

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            double x0 = xc - size/2 + size*i/N;
            double y0 = yc - size/2 + size*j/N;
            Complex z0 = new Complex(x0, y0);
            Color color = mand(z0);
            pic.set(i, N-1-j, color);
        }
    }
    pic.show();
}
```
Mandelbrot Set

% java Mandelbrot -.5 0 2
% java Mandelbrot .1045 -.637 .01
Mandelbrot Set

% java ColorMandelbrot -.5 0 2 < mandel.txt
Mandelbrot Set

(( -1.5, -1 ))
Mandelbrot Set Music Video

http://www.jonathancoulton.com/songdetails/Mandelbrot Set
Data type. Set of values and collection of operations on those values.

Simulating the physical world.
- Java objects model real-world objects.
- Not always easy to make model reflect reality.
- Ex: charged particle, molecule, COS 126 student, ....

Extending the Java language.
- Java doesn't have a data type for every possible application.
- Data types enable us to add our own abstractions.
- Ex: complex, vector, polynomial, matrix, ....
3.2 Extra Slides
Example: Bouncing Ball in Unit Square

**Bouncing ball.** Model a bouncing ball moving in the unit square with constant velocity.
public class Ball {
    private double rx, ry;
    private double vx, vy;
    private double radius;

    public Ball() {
        rx = ry = 0.5;
        vx = 0.015 - Math.random() * 0.03;
        vy = 0.015 - Math.random() * 0.03;
        radius = 0.01 + Math.random() * 0.01;
    }

    public void move() {
        if ((rx + vx > 1.0) || (rx + vx < 0.0)) vx = -vx;
        if ((ry + vy > 1.0) || (ry + vy < 0.0)) vy = -vy;
        rx = rx + vx;
        ry = ry + vy;
    }

    public void draw() {
        StdDraw.filledCircle(rx, ry, radius);
    }
}

---

Example: Bouncing Ball in Unit Square

```java
public class Ball {
    private double rx, ry;
    private double vx, vy;
    private double radius;

    public Ball() {
        rx = ry = 0.5;
        vx = 0.015 - Math.random() * 0.03;
        vy = 0.015 - Math.random() * 0.03;
        radius = 0.01 + Math.random() * 0.01;
    }

    public void move() {
        if ((rx + vx > 1.0) || (rx + vx < 0.0)) vx = -vx;
        if ((ry + vy > 1.0) || (ry + vy < 0.0)) vy = -vy;
        rx = rx + vx;
        ry = ry + vy;
    }

    public void draw() {
        StdDraw.filledCircle(rx, ry, radius);
    }
}
```
Object References

Object reference.

- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
</tr>
<tr>
<td>C3</td>
<td>0</td>
</tr>
<tr>
<td>C4</td>
<td>0</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0</td>
</tr>
<tr>
<td>C8</td>
<td>0</td>
</tr>
<tr>
<td>C9</td>
<td>0</td>
</tr>
<tr>
<td>CA</td>
<td>0</td>
</tr>
<tr>
<td>CB</td>
<td>0</td>
</tr>
<tr>
<td>CC</td>
<td>0</td>
</tr>
</tbody>
</table>

main memory
(64-bit machine)
Object References

Object reference.
- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```
Object Reference

- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```
Object References

Object reference.

- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>0.60</td>
</tr>
<tr>
<td>C1</td>
<td>0.52</td>
</tr>
<tr>
<td>C2</td>
<td>0.05</td>
</tr>
<tr>
<td>C3</td>
<td>0.01</td>
</tr>
<tr>
<td>C4</td>
<td>0.03</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0</td>
</tr>
<tr>
<td>C8</td>
<td>0</td>
</tr>
<tr>
<td>C9</td>
<td>0</td>
</tr>
<tr>
<td>CA</td>
<td>0</td>
</tr>
<tr>
<td>CB</td>
<td>0</td>
</tr>
<tr>
<td>CC</td>
<td>0</td>
</tr>
</tbody>
</table>

main memory (64-bit machine)

registers
Object References

Object reference.
- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>0.60</td>
</tr>
<tr>
<td>C1</td>
<td>0.52</td>
</tr>
<tr>
<td>C2</td>
<td>0.05</td>
</tr>
<tr>
<td>C3</td>
<td>0.01</td>
</tr>
<tr>
<td>C4</td>
<td>0.03</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0.50</td>
</tr>
<tr>
<td>C8</td>
<td>0.50</td>
</tr>
<tr>
<td>C9</td>
<td>0.07</td>
</tr>
<tr>
<td>CA</td>
<td>0.04</td>
</tr>
<tr>
<td>CB</td>
<td>0.04</td>
</tr>
<tr>
<td>CC</td>
<td>0</td>
</tr>
</tbody>
</table>

main memory (64-bit machine)

registers
**Object References**

- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>0.60</td>
</tr>
<tr>
<td>C1</td>
<td>0.52</td>
</tr>
<tr>
<td>C2</td>
<td>0.05</td>
</tr>
<tr>
<td>C3</td>
<td>0.01</td>
</tr>
<tr>
<td>C4</td>
<td>0.03</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0.57</td>
</tr>
<tr>
<td>C8</td>
<td>0.54</td>
</tr>
<tr>
<td>C9</td>
<td>0.07</td>
</tr>
<tr>
<td>CA</td>
<td>0.04</td>
</tr>
<tr>
<td>CB</td>
<td>0.04</td>
</tr>
<tr>
<td>CC</td>
<td>0</td>
</tr>
</tbody>
</table>

Registers (64-bit machine)
Object References

Object reference.
- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();
b2 = b1;
b2.move();

Data stored in C7 - CB for abstract bit recycler.

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>0.60</td>
</tr>
<tr>
<td>C1</td>
<td>0.52</td>
</tr>
<tr>
<td>C2</td>
<td>0.05</td>
</tr>
<tr>
<td>C3</td>
<td>0.01</td>
</tr>
<tr>
<td>C4</td>
<td>0.03</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0.57</td>
</tr>
<tr>
<td>C8</td>
<td>0.54</td>
</tr>
<tr>
<td>C9</td>
<td>0.07</td>
</tr>
<tr>
<td>CA</td>
<td>0.04</td>
</tr>
<tr>
<td>CB</td>
<td>0.04</td>
</tr>
<tr>
<td>CC</td>
<td>0</td>
</tr>
</tbody>
</table>

main memory
(64-bit machine)
Object References

Object reference.
- Allow client to manipulate an object as a single entity.
- Essentially a machine address (pointer).

```java
Ball b1 = new Ball();
b1.move();
b1.move();

Ball b2 = new Ball();
b2.move();

b2 = b1;
b2.move();
```

Moving `b2` also moves `b1` since they are aliases that reference the same object.
Creating Many Objects

Each object is a data type value.

- Use `new` to invoke constructor and create each one.
- Ex: create N bouncing balls and animate them.

```java
public class BouncingBalls {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        Ball balls[] = new Ball[N];
        for (int i = 0; i < N; i++)
            balls[i] = new Ball();

        while (true) {
            StdDraw.clear();
            for (int i = 0; i < N; i++) {
                balls[i].move();
                balls[i].draw();
            }
            StdDraw.show(20);
        }
    }
}
```
50 Bouncing Balls

**Color.** Associate a color with each ball; paint background black.

```bash
% java BouncingBalls 50
```

**Scientific variations.** Account for gravity, spin, collisions, drag, ...
OOP Context

Reference. Variable that stores the name of a thing.

<table>
<thead>
<tr>
<th>Thing</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web page</td>
<td><a href="http://www.princeton.edu">www.princeton.edu</a></td>
</tr>
<tr>
<td>Bank account</td>
<td>45-234-23310076</td>
</tr>
<tr>
<td>Word of TOY memory</td>
<td>1C</td>
</tr>
<tr>
<td>Byte of computer memory</td>
<td>00FACADE</td>
</tr>
<tr>
<td>Home</td>
<td>35 Olden Street</td>
</tr>
</tbody>
</table>

Some consequences.
- Assignment statements copy references (not objects).
- The == operator tests if two references refer to same object.
- Pass copies of references (not objects) to functions.
  - efficient since no copying of data
  - function can change the object
Client. A sample client program that uses the Point data type.

```java
public class PointTest {
    public static void main(String[] args) {
        Point a = new Point();
        Point b = new Point();
        double distance = a.distanceTo(b);
        StdOut.println("a = " + a);
        StdOut.println("b = " + b);
        StdOut.println("distance = " + distance);
    }
}
```

% java PointTest
a = (0.716810971264761, 0.0753539063358446)
b = (0.4052136795358151, 0.033848435224524076)
distance = 0.31434944941098036
Points in the Plane

Data type. Points in the plane.

```java
public class Point {
    private double x;
    private double y;

    public Point() {
        x = Math.random();
        y = Math.random();
    }

    public String toString() {
        return "(" + x + ", " + y + ")";
    }

    public double distanceTo(Point p) {
        double dx = x - p.x;
        double dy = y - p.y;
        return Math.sqrt(dx*dx + dy*dy);
    }
}
```

\[ \sqrt{dx^2 + dy^2} \]
A Compound Data Type: Circles

**Goal.** Data type for circles in the plane.

```java
public class Circle {
    private Point center;
    private double radius;

    public Circle(Point center, double radius) {
        this.center = center;
        this.radius = radius;
    }

    public boolean contains(Point p) {
        return p.dist(center) <= radius;
    }

    public double area() {
        return Math.PI * radius * radius;
    }

    public boolean intersects(Circle c) {
        return center.dist(c.center) <= radius + c.radius;
    }
}
```
Pass-By-Value

Arguments to methods are always passed by value.

- Primitive types: passes copy of value of actual parameter.
- Objects: passes copy of reference to actual parameter.

```java
public class PassByValue {
    static void update(int a, int[] b, String c) {
        a = 7;
        b[3] = 7;
        c = "seven";
        StdO.println(a + " " + b[3] + " " + c);
    }

    public static void main(String[] args) {
        int a = 3;
        int[] b = { 0, 1, 2, 3, 4, 5 };
        String c = "three";
        StdOut.println(a + " " + b[3] + " " + c);
        update(a, b, c);
        StdOut.println(a + " " + b[3] + " " + c);
    }
}
```

% java PassByValue
3 3 three
7 7 seven
3 7 three