

3.2: Creating Data Types

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

Built-in "primitive" types.

Data Type	Set of Values	Some Operations
boolean	true, false	not, and, or, xor
int	any of 2^{32} possible integers	add, subtract, multiply
String	any sequence of characters	concatenate, compare

Last time: write programs that use custom data types.

Today: write programs to create custom data types.

Defining Data Types in Java

To define a data type, define:

- Set of values.
- Operations defined on them.

A Java class allows us to define data types by specifying:

- A set of variables. (set of values)
- Methods. (operations defined on them)
- Constructors. (for creating and initializing new objects)

Example: Bouncing Ball in Unit Square

```

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.go(rx, ry);
        StdDraw.spot(2 * radius);
    }
}

```

Annotations in the code block:
 - **data members**: points to the private double variables.
 - **(a Ball is characterized by 5 real numbers)**: comment describing the data members.
 - **constructor**: points to the Ball() method.
 - **bounce** and **method 1**: points to the move() method.
 - **method 2**: points to the draw() method.

Using Data Types in Java

A client is a program that uses a data type.

- Create *objects* with `new`.
- Invoke *methods* with dot operator to perform operations.

```
public class BallDemo {
    public static void main(String[] args) {
        Ball b = new Ball();
        StdDraw.create(512, 512);
        StdDraw.setScale(0, 0, 1, 1);
        while (true) {
            StdDraw.clear();
            b.move();
            b.draw();
            StdDraw.pause(10);
        }
    }
}
```

5

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();
```

addr	value
C0	0
C1	0
C2	0
C3	0
C4	0
C5	0
C6	0
C7	0
C8	0
C9	0
CA	0
CB	0
CC	0

main memory
(64-bit machine)

6

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();
```

addr	value
C0	0.50
C1	0.50
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0
C8	0
C9	0
CA	0
CB	0
CC	0

registers

main memory
(64-bit machine)

7

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();
```

addr	value
C0	0.55
C1	0.51
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0
C8	0
C9	0
CA	0
CB	0
CC	0

registers

main memory
(64-bit machine)

8

Object References

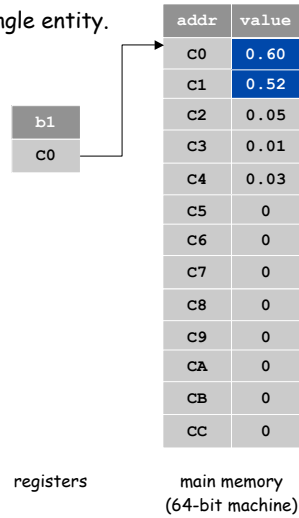
Object reference.

- 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 ();
```



9

Object References

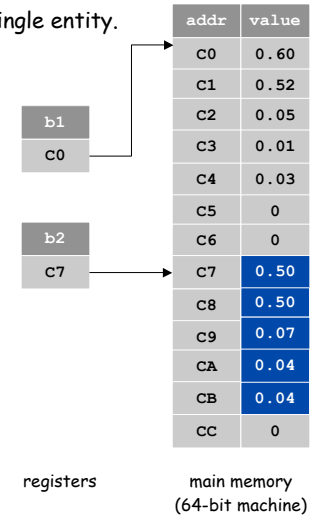
Object reference.

- 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 ();
```



10

Object References

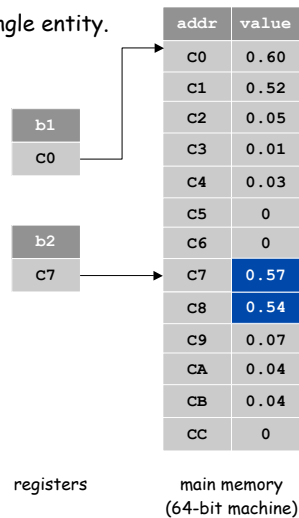
Object reference.

- 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 ();
```



11

Object References

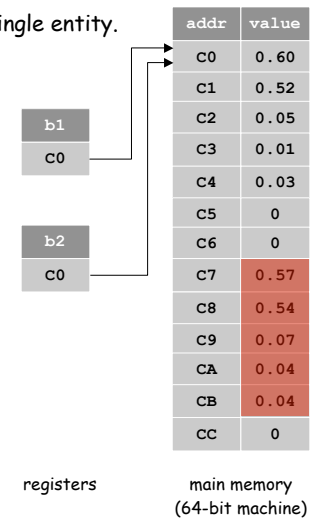
Object reference.

- 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.



12

Object References

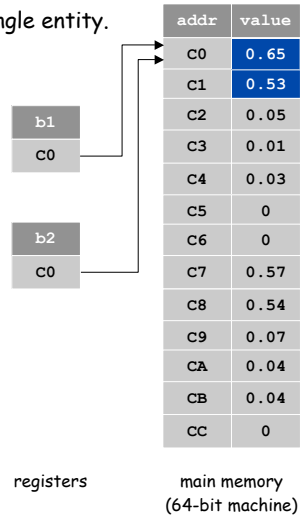
Object reference.

- 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 ();
```



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

OOP Context

Reference. Variable that stores the name of a thing.

Thing	Name
Web page	www.princeton.edu
Bank account	45-234-23310076
Word of TOY memory	1C
Byte of computer memory	00FACADE
Home	35 Olden Street

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

References

René Magritte. "This is not a pipe."



Java. This is not a bouncing ball.

```
Ball b = new Ball ();
b.move ();
b.show ();
```

OOP. Natural vehicle for studying abstract models of the real world.

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.

```
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 ();

        StdDraw.create(512, 512);
        while(true) {
            StdDraw.clear();
            for (int i = 0; i < N; i++) {
                balls[i].move ();
                balls[i].draw ();
            }
            StdDraw.pause(20);
        }
    }
}
```

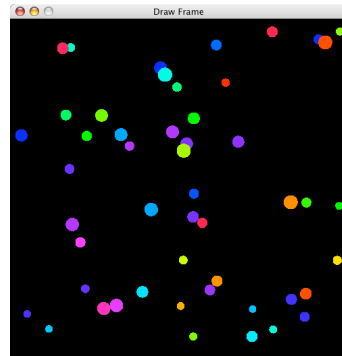
create and initialize N objects

animation loop

50 Bouncing Balls

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

```
% java BouncingBalls 50
```



Scientific variations. Account for gravity, spin, collisions, drag,

A Database Example

Student record: first name, last name, email address, section number.

```
public class Student {
    private String first;
    private String last;
    private String email;
    private int section;

    Student(String first, String last, String email, int section) {
        this.first = first;
        this.last = last;
        this.email = email;
        this.section = section;

    }

    public boolean less(Student x) { return section < x.section; }

    public String toString() {
        return section + " " + first + " " + last + " " + email;
    }
}
```

Annotations in the original image:

- ← data members (pointing to the private fields)
- can pass parameters to constructor (pointing to the constructor parameters)
- this = reference to object just created (pointing to the 'this' assignments)
- is invoking object's section # less than x's? (pointing to the less method)
- returns a String representation of a student (pointing to the toString method)

18

19

Sorting by Section Number

```
public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    Student[] s = new Student[N];

    for (int i = 0; i < N; i++) {
        String first = StdIn.readString();
        String last = StdIn.readString();
        String email = StdIn.readString();
        int section = StdIn.readInt();
        s[i] = new Student(first, last, email, section);
    }

    for (int i = 0; i < N; i++)
        for (int j = i; j > 0; j--)
            if (s[j].less(s[j-1])) {
                Student swap = s[j];
                s[j] = s[j-1];
                s[j-1] = swap;
            }

    for (int i = 0; i < N; i++)
        System.out.println(s[i]);
}
```

Annotations in the original image:

- read in data (pointing to the first loop)
- insertion sort (pointing to the second loop)
- compare section # (pointing to the if condition)
- swap references (pointing to the swap assignments)
- print results (pointing to the third loop)
- invokes toString method automatically (pointing to the println call)

20

Sample Output

```
% more students.txt
Austin Taylor actaylor 1
David Rosner drosner 4
Rebecca Allen rebecca 7
Rajiv Ayyangar ayyangar 7
Daniel Barrett drbarret 8
Nic Byrd nbyrd 7
Emily Capra ecapra 8
Johnny Clore jclore 7
Peter Combs pcombs 8
Chris D'Ambrosia cdambros 8
Andrew Ferguson owsla 7
Richa Gawande rgawande 8
Sam Grossberg sgrossbe 7
Anita Gupta aagupta 8
Weiyin He weiyinhe 8
Douglas Hohensee dgh 7
Mallory James mgjames 8
. . .
Mateusz Plucinski mplucins 5
Robert Krone rkrone 3
Yana Krasteva krasteva 5
Zhen Xia zxia 2
```

```
% java Students 127 < students.txt
1 Austin Taylor actaylor
1 Thomas Arias tarias
1 Nikola Kamburov kamburov
1 Arti Sheth asheth
1 Abraham Bassan abassan
1 Mary Bobrowski mbobrows
1 Ryan Corces-Zimmerman mcorces
1 Pierre-Etienne Genest pgenest
1 Lauren Lichtman llichtma
1 Michael Mashiba mmashiba
1 Tara Lloyd tlloyd
1 Lawrence Azzaretti lazzaret
1 Manisha Bhattacharya mbhattac
1 Matt Kopko mkopko
1 Eric Miller elmiller
1 Ishani Sud isud
2 Loan Le lle
. . .
8 Erika Sloan esloan
8 Hanlin Tang hanlint
8 Mary Wathall walthall
8 Sharon Weeks sweets
```

21

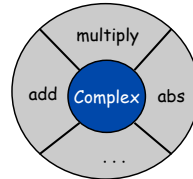
Complex Number Data Type

Goal. Extend the Java language by creating a data type to manipulate complex numbers.

Set of values. Two real numbers (real and imaginary parts).

Set of operations.

- Add, subtract, multiply, divide.
- Absolute value, phase.
- Convert to string.



Applications.

- Fractals.
- Impedance in RLC circuits.
- Signal processing and Fourier analysis.
- Control theory and Laplace transforms.
- Quantum mechanics and Hilbert spaces.

Complex Number Data Type: A Sample Client

Client program uses data type operations to calculate something.

```
public static void main(String[] args) {
    Complex a = new Complex(5.0, 6.0);
    System.out.println("a = " + a);

    Complex b = new Complex(-2.0, 3.0);
    System.out.println("b = " + b);

    Complex c = a.times(b);
    System.out.println("c = " + c);
}
```

System knows how to print a Complex object

```
% java TestClient
a = 5.0 + 6.0i
b = -2.0 + 3.0i
c = -28.0 + 3.0i
```

$$(5 + 6i) * (-2 + 3i) = -28 + 3i$$

Remark: can't write `c = a * b` since no operator overloading in Java.

22

23

Complex Number Data Type: Implementation

```
public class Complex {
    private double re;
    private double im;
```

← data members

```
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);
}
```

Complex Number Data Type: Implementation (cont)

```
public class Complex {
    ...
}
```

creates a Complex object, and returns a reference to it

```
public Complex plus(Complex b) {
    Complex a = this;
    Complex c = new Complex(0, 0);
    c.re = a.re + b.re;
    c.im = a.im + b.im;
    return c;
}
```

$$(5 + 6i) + (-2 + 3i) = 3 + 9i$$

```
public Complex times(Complex b) {
    Complex a = this;
    Complex c = new Complex(0, 0);
    c.re = a.re * b.re - a.im * b.im;
    c.im = a.re * b.im + a.im * b.re;
    return c;
}
```

$$(5 + 6i) * (-2 + 3i) = (-10 - 18) + (15 - 12)i = -28 + 3i$$

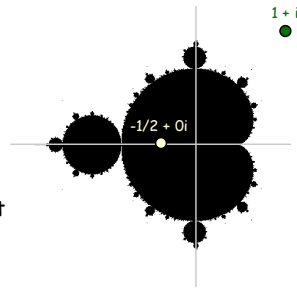
24

25

Mandelbrot Set

Mandelbrot set.

- Set of points in the plane.
- To check if $z = x + iy$ is in set, initialize $c_0 = z$ and iterate $c_{t+1} = (c_t)^2 + z$.
 - if $|c_t|$ diverges to infinity, then z not in set
 - otherwise z is in set



i	c_i
0	$1 + i$ ●
1	$1 + 3i$
2	$-7 + 7i$
3	$1 - 97i$
4	$-9407 - 193i$
5	$88454401 + 3631103i$

$z = 1 + i$ not in Mandelbrot set

i	c_i
0	$-1/2$ ○
1	$-1/4$
2	$-7/16$
3	1
4	$-79/256$
5	$-26527/65536$

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

26

Plotting the Mandelbrot Set

Plot. Color (x, y) black if $z = x + iy$ is in the set, and white otherwise.

Practical issues.

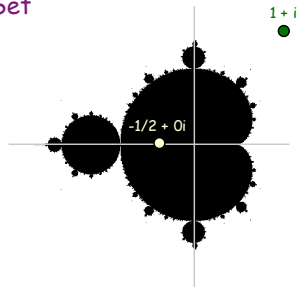
- Cannot plot infinitely many points.
- Cannot iterate infinitely many times.

Approximate solution.

- Choose a finite grid of points in the plane.
- Iterate Mandelbrot function for at most 256 iterates.
 - fact: if $|c_t| > 2$ for any t , then z not in Mandelbrot set
 - pseudo-fact: if $|c_{256}| \leq 2$ then z "likely" in Mandelbrot set

More dramatic picture.

- Plot z in color according to number of iterates until $|c_t| > 2$.



27

Complex Number Data Type: Another Client

Mandelbrot function with complex numbers.

- Is z in the Mandelbrot set?
- Returns an integer between 0 and 255.

```
public static int mand(Complex z) {
    Complex c = z;
    for (int t = 0; t < 255; t++) {
        if (c.abs() >= 2.0) return t;
        c = c.times(c);
        c = c.plus(z);
    }
    return 255;
}
```

$c = c * c + z$

- 7 lines of code with judicious use of data types.

28

Complex Number Data Type: Another Client

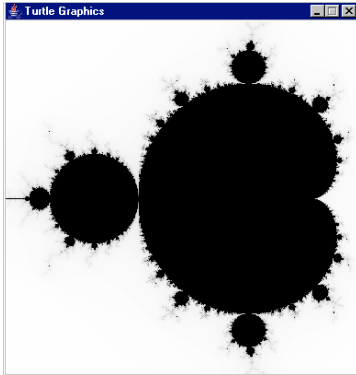
Plot the Mandelbrot set in gray scale.

```
public static void main(String args[]) {
    double xmin = Double.parseDouble(args[0]);
    double ymin = Double.parseDouble(args[1]);
    double width = Double.parseDouble(args[2]);
    double height = Double.parseDouble(args[3]);
    double SIZE = 512;

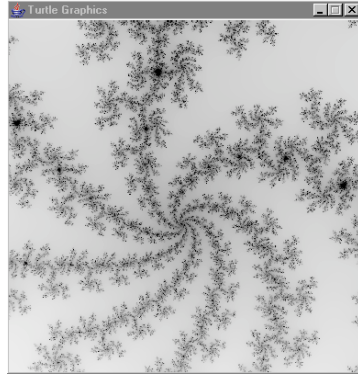
    Picture pic = new Picture(size, size);
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            double x = xmin + (i * width / SIZE);
            double y = ymin + height - (j * height / SIZE);
            Complex z = new Complex(x, y);
            int c = 255 - mand(z);
            pic.setGray(c, i, j);
        }
    }
    pic.show();
}
```

29

Mandelbrot Set



```
% java Mandelbrot -1.5 -1 2 2
```



```
% java Mandelbrot .10259 -.641 .0086 .0086
```

Applications of Data Types

Data type: set of values and collection of operations on those values.

Simulating the physical world.

- Java objects model real-world objects.
- Ex: bouncing ball, *COS 126* student.
- Not always easy to make model reflect reality: collisions, gravity.

Extending the Java language.

- Java doesn't have a data type for every possible application.
- Data types enable us to address other mathematical abstractions.
- Scientific applications: complex numbers, polynomials, matrices,