Lecture 13: Creating Data Types

Data Types

Data type.
- Set of values and operations on those values.
- There are a few built-in "primitive" types, e.g.:

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

We want to write programs that process other types of data.
- Complex numbers, matrices, lists, playing cards, sound waves . . .

So today: we write programs to create and manipulate 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 {                                    Ball.java

    private double rx, ry;        ⬅ data members
    private double vx, vy;           (a Ball is characterized by 5 real numbers)
    private double radius;

    public Ball() {                                    constructor
        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;                                  bounce
    }                                                  method 1

    public void draw() {
        StdDraw.go(rx, ry);
        StdDraw.spot(2 * radius);
    }                                                  method 2
}
```

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.

```java
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);
        }
    }
}
```

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();
```

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)

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();
```

b1
C0

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
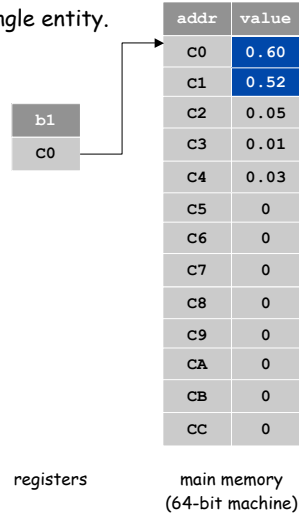(64-bit machine)

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();
```

b1
C0

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

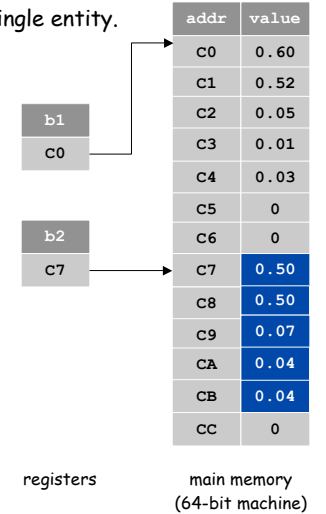registers   main memory
(64-bit machine)

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();
```

registers

b1
C0

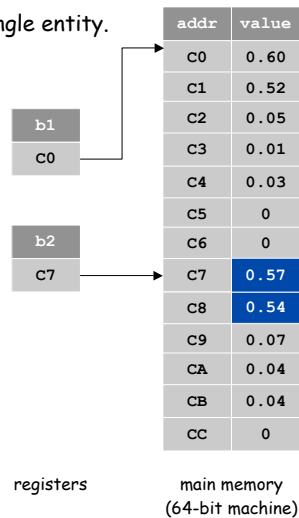| addr	value
C0	0.60
C1	0.52
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0
C8	0
C9	0
CA	0
CB	0
CC	0

main memory (64-bit machine)

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();
```

registers

b1
C0

b2
C7

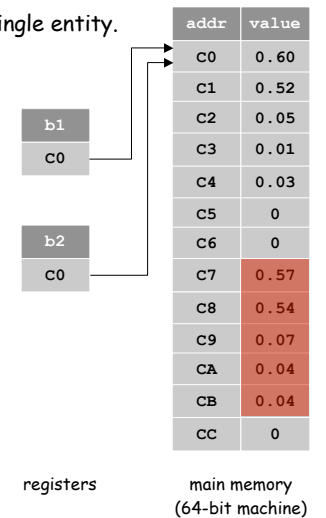| addr	value
C0	0.60
C1	0.52
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0.50
C8	0.50
C9	0.07
CA	0.04
CB	0.04
CC	0

main memory (64-bit machine)

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();
```

registers

b1
C0

b2
C7

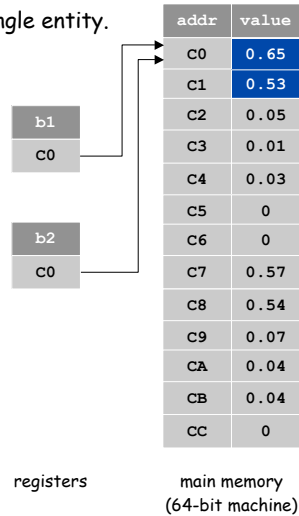| addr	value
C0	0.60
C1	0.52
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0.57
C8	0.54
C9	0.07
CA	0.04
CB	0.04
CC	0

main memory (64-bit machine)

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.

registers

b1
C0

b2
C0

addr	value
C0	0.60
C1	0.52
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0.57
C8	0.54
C9	0.07
CA	0.04
CB	0.04
CC	0

main memory (64-bit machine)

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();
```

addr	value
C0	0.65
C1	0.53
C2	0.05
C3	0.01
C4	0.03
C5	0
C6	0
C7	0.57
C8	0.54
C9	0.07
CA	0.04
CB	0.04
CC	0

b1 → C0

b2 → C0

registers

main memory
(64-bit machine)

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

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]);        create and initialize
        Ball balls[] = new Ball[N];               N objects
        for (int i = 0; i < N; i++)
            balls[i] = new Ball();

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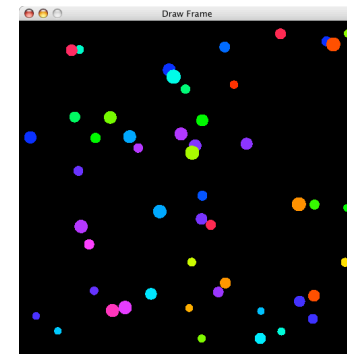
---

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.

```java
public class Student {                            Student.java
    private String first;          ← data members
    private String last;
    private String email;
    private int section;
                          can pass parameters to constructor

    Student(String first, String last, String email, int section) {
        this.first   = first;
        this.last    = last;          this = reference to object just created
        this.email   = email;
        this.section = section;
    }
                          is invoking object's section # less than x's?

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

    public String toString() {
        return section + " " + first + " " + last + " " + email;
    }
}
                          returns a String representation of a student
```

Sorting by Section Number

```java
public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);          read in data
    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++)                  insertion sort
        for (int j = i; j > 0; j--)
            if (s[j].less(s[j-1])) {      ← compare section #
                Student swap = s[j];
                s[j] = s[j-1];            ← swap references
                s[j-1] = swap;
            }

    for (int i = 0; i < N; i++)                  print results
        System.out.println(s[i]);
}
                          invokes toString method automatically
```

Sample Output

```
% more students.txt
Austin Taylor actaylor 1
David Rosner drosner 4
Rebecca Allen rebeccaa 7
Rajiv Ayyangar ayyangar 7
Daniel Barrett drbarrett 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 sweeks
```
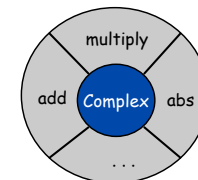
Complex Number Data Type

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.

```java
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.

Complex Number Data Type: Implementation

```java
public class Complex {
   private double re;        ⬅ data members
   private double im;

   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)

```java
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;
   }

   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;
   }
}
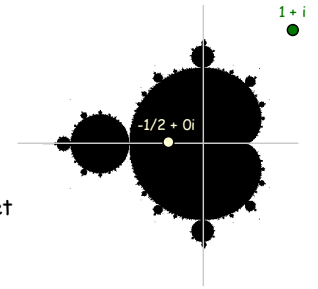```

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

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

---

## Mandelbrot Set

1 + i

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

-1/2 + 0i

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

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

z = 1 + i not in Mandelbrot set

z = -1/2 is in Mandelbrot set

## Plotting the Mandelbrot Set

1 + i

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

-1/2 + 0i

### 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 $|c_{256}| \le 2$ then z "likely" in Mandelbrot set

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

---

## Complex Number Data Type:  Another Client

**Mandelbrot function with complex numbers.**
- Is z in the Mandelbrot set?
- Returns an integer between 0 and 255.

```java
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);                    c = c * c + z
    }
    return 255;
}
```

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

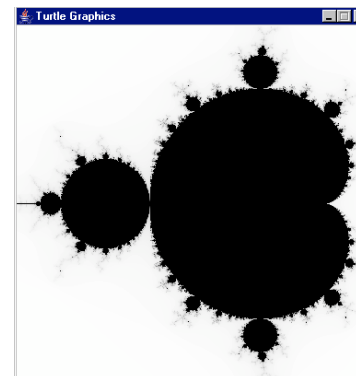## Complex Number Data Type: Another Client

Plot the Mandelbrot set in gray scale.

```java
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;

    StdDraw.create(SIZE, SIZE);              scale to screen
    for (int i = 0; i < SIZE; i++) {         coordinates
        for (int j = 0; j < SIZE; j++) {
            double x = xmin +       (i * width  / SIZE);
            double y = ymin +       (j * height / SIZE);
            Complex z = new Complex(x, y);
            int c = 255 - mand(z);
            StdDraw.setColor(new Color(c, c, c));
            StdDraw.go(i, j); StdDraw.spot();     grayscale
        }
    }
    StdDraw.show();
}
```
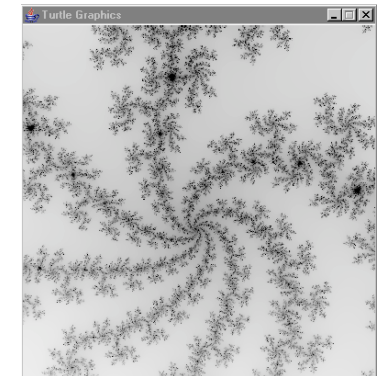

Mandelbrot Set

% 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, ...

Announcements

Thinking about majoring in Computer Science?

Or doing the Certificate in Applications of Computing?

Then: visit the all-new "Life in the Computer Science Department: A Guide for the Humble Undergraduate":
- http://www.cs.princeton.edu/ugradpgm/life.html
- a handy FAQ that answers many many questions

And/Or: Come talk to me

AND CERTAINLY attend at least one of:
- C.S. open house for BSE freshmen Tuesday 3/29, Friend Convocation Room, 5:45 (PM!): tours, demos, pizza (AB's welcome)
- C.S. open house for AB sophomores Tuesday April 5, C.S. Tea Room, 4 PM (but no pizza, and maybe fewer demos) (BSE's welcome)

Announcements

Not-exactly Midterm Exam
- Tomorrow night, Wed March 23, 7:30 PM, right here
- Closed book, but
- You can bring one *cheatsheet*
   - one side of one (8.5 by 11) sheet, handwritten by you
- P.S. No calculators, laptops, cellphones, etc.
- Covers first half of course
- If TOY programming is required, a TOY cheatsheet will be provided

Helpful review session
- Tonight, Tuesday March 22, 7:30 PM, COS 105
- Not a canned presentation
- Driven by your questions

Clark's Principles of exam preparation:
- 1) Practice! (don't just read)   2) Write your own exam   3) Study in groups