Lecture 13: Data Types

Data Types

Data type.
- Set of values and operations on those values.
- There are a few 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

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

To define a data type, define:
- Set of values.
- Operations defined on them.

Defining Data Types in Java

A Java `class` allows us to define data types by specifying:
- A set of variables. (a set of values)
- A set of functions. (a set of "methods" on those values)
- Ex: bouncing ball in the unit square.

Ball.java

```
public class Ball {

   private double px, py;
   private double vx, vy;

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

   public void draw(double w, double h) {
      Turtle.fly(w * px, h * py);
      Turtle.spot(20);
   }
}
```

⇐ data members (a `Ball` is characterized 4 real numbers)

methods (two allowed operations)

bounce

Using Data Types in Java

A client is a program that uses a data type.
- Create *objects* (variables having one of the values) with `new`.
- Invoke *methods* to perform operations on objects.
- Clientness is relative.

```
public static void main(String[] args) {
   Ball b = new Ball();
   b.px = 0.5;
   b.py = 0.5;
   b.vx = 0.03;
   b.vy = 0.02;

   Turtle.create(512, 512);
   while (true) {
      Turtle.clear();
      b.move();
      b.draw(512, 512);
      Turtle.pause(10);
   }
}
```

creates a new object of type `Ball`

⇐ use . to access data members (usually do NOT refer to data in client)

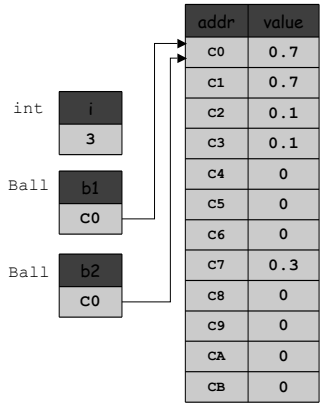
⇐ use . to invoke a method

Object References

Object reference.

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

```
int i = 3;

Ball b1 = new Ball();
b1.px = 0.5;
b1.py = 0.5;
b1.vx = 0.1;
b1.vy = 0.1;
b1.move();

Ball b2 = new Ball();
b2.px = 0.3;
b2 = b1;
b2.move();
```



Moving `b2` also moves `b1` since they point to the same memory address.

Creating Objects

Constructor.

- Creates an object and returns a *reference* to the object.
- Client invokes by using keyword `new`.
- Typically initializes values.

```
public class Ball {
   private double px, py;
   private double vx, vy;

   public Ball() {
      px = 0.5;
      py = 0.5;
      vx = 0.015 - Math.random() * 0.03;
      vy = 0.015 - Math.random() * 0.03;
   }

   // move and draw methods go here

}
```

⇐ constructor looks like a method with same name as class, but does not have a return type

Ball.java

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.

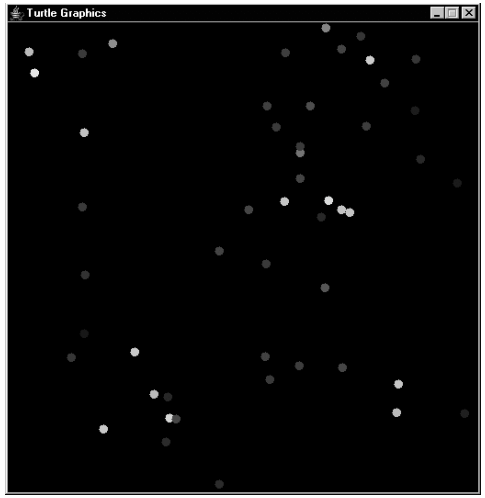
Balls.java

```
public class Balls {
   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();

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

create and initialize N objects

animation loop

50 Bouncing Balls




```
% java Balls 50
```

Another Example

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

Student.java

```
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;
   }
}
```

⇐ data members

can pass parameters to constructor

this = reference to object just created

is invoking object's section # less than x's?

returns a String representation of a student

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]);
}
```

read in data

insertion sort

⇐ compare section #

⇐ swap references

print results

invokes `toString` method automatically

Sample Output

```
% more students.txt
Abraham Flamholz flamholz 3
Aditi Shrivastava ashrivas 4
Alexander Thorn athorn 2
Alicia Myers amyers 1
Amy Trangsrud atrangsr 1
Anand Dharan adharan 1
Anshuman Sahoo asahoo 3
Arthur Shum ashum 1
Arti Sheth asheth 5
Ashley Evans amevans 1
Avi Ziskind aziskind 5
Benjamin Amster bamster 2
Bryant Chen bryantc 1
Cameron Brien cbrien 3
Caroline Teichner cteichne 3
Charles Alden calden 1
Cole Deforest cde 1
Daniel Potter dpotter 3
Darin Sleiter dsleiter 3
David Astle dastle 1
. . .
Leizhi Sun leizhis 3
Lester MacKey lmackey 3
```

```
% java Students 79 < students.txt
1 Alicia Myers amyers
1 Amy Trangsrud atrangsr
1 Anand Dharan adharan
1 Arthur Shum ashum
1 Ashley Evans amevans
1 Bryant Chen bryantc
1 Charles Alden calden
1 Cole Deforest cde
1 David Astle dastle
1 Elinor Keith ekeith
1 John Kim johnkim
1 Josh Probst jprobst
1 Julia Ressler jressler
1 Kira Hohensee hohensee
1 Maria Miguez mmiguez
1 Michael Reilly mcreilly
1 Nancy Khov nkhov
1 Tarik Jones tarikj
2 Alexander Thorn athorn
2 Benjamin Amster bamster
. . .
5 Tom Brennan tpbrenna
5 Yiting Jin ycjin
```


Complex Number Data Type

Extending the Java language.

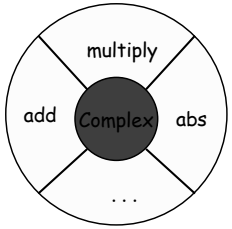
- Create a user-defined type to manipulate complex numbers.

Set of values.

- Two real numbers (real and imaginary parts).

Set of operations.

- Initialize.
- Copy. ← Recall: assignment statement with objects behaves differently than with primitive types.
- Add
- Multiply.
- Absolute value (magnitude).
- Convert to string.
- Subtract, divide, complex conjugate, . . .



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 = Complex.multiply(a, 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
```

no operator overloading in Java: can't write c = a * b

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

Complex Number Data Type: Implementation

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

   Complex(double re, double im) { this.re = re;   this.im = im;   }

   public String toString() { return re + " + " + im + "i"; }

   public static Complex add(Complex a, Complex b) {
      return new Complex(a.re + b.re, a.im + b.im);
   }

   public Complex multiply(Complex a, Complex b) {
      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;
   }

   public double abs() { return Math.sqrt(re*re + im*im); }
```

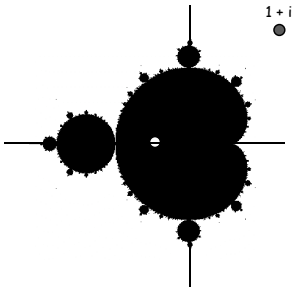
data members

constructors

copy

method creates and returns a Complex object

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

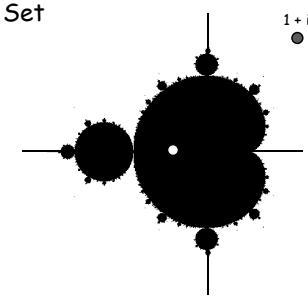
i	c_i
0	1 + i
1	1 + 3i
2	-7 + 7i
3	1 - 97i
4	-9407 - 193i
5	88454401 + 3631103i

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

Plotting the Mandelbrot Set



Plot (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 $|c_{256}| \le 2$ 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.

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

c = c * c + z

a function that takes an object as input and returns an integer

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

Complex Number Data Type: Another Client

Plot the Mandelbrot set in gray scale.

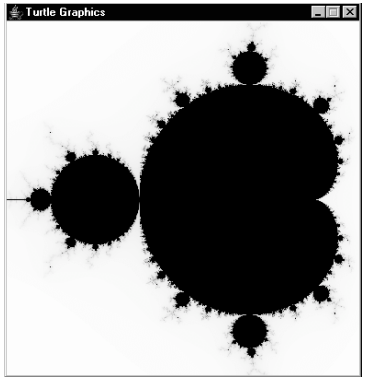
```
public static void main(String args[]) {
   double xmin   = Double.parseDouble(args[0]);
   double ymin   = Double.parseDouble(args[1]);
   double width  = Double.parseDouble(args[2]);
   double height = Double.parseDouble(args[3]);

   Turtle.create(512, 512);
   for (int i = 0; i < 512; i++) {
      for (int j = 0; j < 512; j++) {
         double x = xmin + i * width  / 512;
         double y = ymin + j * height / 512;
         Complex z = new Complex(x, y);
         int c = 255 - mand(z);
         Turtle.setColor(new Color(c, c, c));
         Turtle.pixel(i, j);
      }
   }
   Turtle.destroy();
}
```

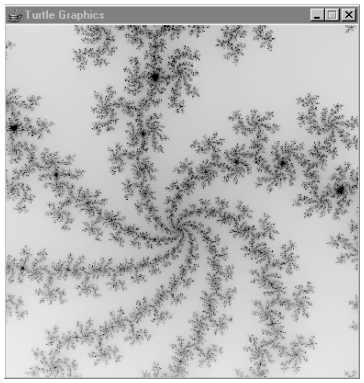
scale to screen coordinates

plots a single pixel

grayscale

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, ...