Object Orientation

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Object Oriented Programming

- a revolutionary extension of programming
- extends earlier programming abstractions
- is the leading programming paradigm
- similar to techniques of thinking about problems in other domains e.g. architecture

Object Oriented Programming

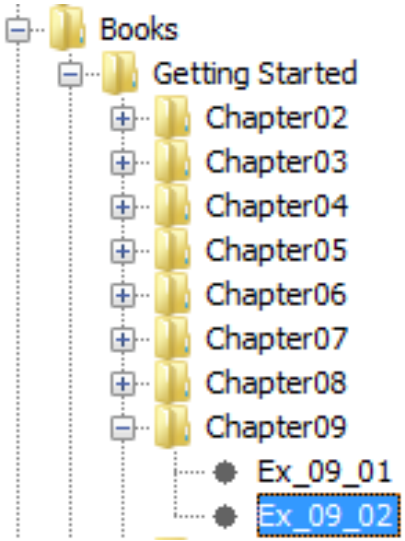
Program consist of many “things” (objects)

- **there are different kinds of “things”**
- **objects are created as instances of classes**
- **objects can have an internal state and components.**
- **objects exchange messages**
- **if object A sends message to B then B does something (and returns a result to A)**
- **results can be boolean, int, float or string**
- **or they can be an object themselves or there is no result (void).**
- **there is some main object with a loop that starts everything off**

Encapsulation

- **objects encapsulate state as a collection of instance variables**
- **objects encapsulate behavior via methods invoked by messages**

JitBug
float x float y int diameter float speed
void move() void display()



Encapsulation

- creating for objects with encapsulated state
- and encapsulated behaviour
- hiding implementation details
- protecting the state information of objects
- putting objects in control
- facilitating modularity, code reuse and maintenance

Encapsulation

- creating for objects with encapsulated state
- and encapsulated behaviour
- hiding implementation details
- protecting the state information of objects
- putting objects in control
- facilitating modularity, code reuse and maintenance

```
JitterBug jit;
JitterBug bug;

void setup() {
  size(480, 120);
  smooth();
  jit = new JitterBug(width * 0.33, height/2, 50);
  bug = new JitterBug(width * 0.66, height/2, 10);
}

void draw() {
  jit.move();
  jit.display();
  bug.move();
  bug.display();
}
```

Class definition

```
class JitterBug {

  float x;
  float y;
  int diameter;
  float speed = 2.5;

  JitterBug(float tempX, float tempY, int tempDiameter) {
    x = tempX;
    y = tempY;
    diameter = tempDiameter;
  }

  void move() {
    x += random(-speed, speed);
    y += random(-speed, speed);
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }
}
```

state attributes

constructor

methods

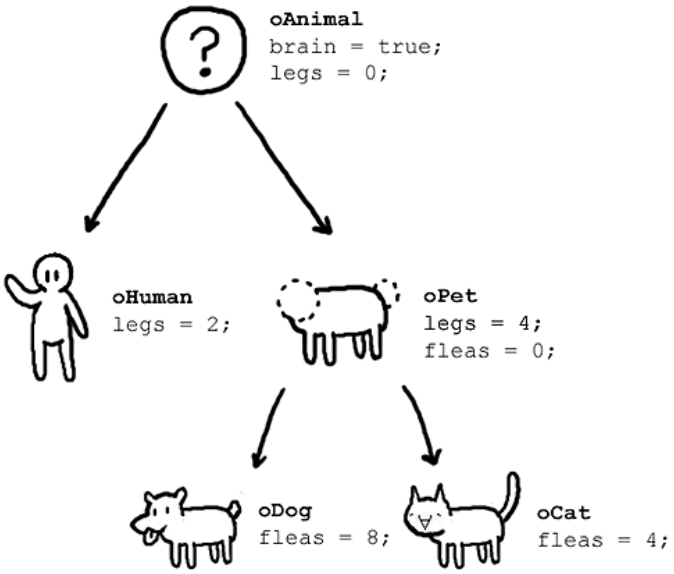
Challenge: add a third jitbug

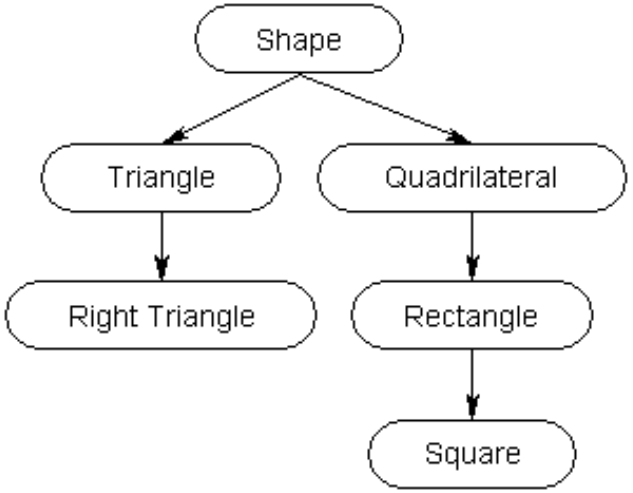
```
JitterBug jit;
JitterBug bug;

void setup() {
  size(480, 120);
  smooth();
  jit = new JitterBug(width * 0.33, height/2, 50);
  bug = new JitterBug(width * 0.66, height/2, 10);
}

void draw() {
  jit.move();
  jit.display();
  bug.move();
  bug.display();
}
```

```
class JitterBug {

  float x;
  float y;
  int diameter;
  float speed = 2.5;

  JitterBug(float tempX, float tempY, int tempDiameter) {
    x = tempX;
    y = tempY;
    diameter = tempDiameter;
  }

  void move() {
    x += random(-speed, speed);
    y += random(-speed, speed);
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }
}
```

Inheritance / Extension

- **Classes form a hierarchy**
  - **superclass is the parent and subclass is a child**
  - **subclasses “extend” (i.e. specialize) their superclass**





ColorJitBug extends JitBug

```
class JitterBug {

  float x;
  float y;
  int diameter;
  float speed = 2.5;

  JitterBug(float tempX, float tempY, int tempDiameter) {
    x = tempX;
    y = tempY;
    diameter = tempDiameter;
  }

  void move() {
    x += random(-speed, speed);
    y += random(-speed, speed);
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }
}
```

```
class ColorJitterBug extends JitterBug {
  color c;

  ColorJitterBug(float tempX, float tempY, int tempDiameter) {
    super(tempX, tempY, tempDiameter);
    c = color(int(random(255)), int(random(255)), int(random(255)));
  }

  void display() {
    pushStyle();
    fill(c);
    super.display();
    popStyle();
  }
}
```

ColorJitBug extends JitBug

```
JitterBug jit;
ColorJitterBug bug;

void setup() {
  size(480, 120);
  smooth();
  jit = new JitterBug(width * 0.33, height/2, 50);
  bug = new ColorJitterBug(width * 0.66, height/2, 10);
}

void draw() {
  jit.move();
  jit.display();
  bug.move();
  bug.display();
}
```

Object Orientation

- **Encapsulation**
- **Inheritance**