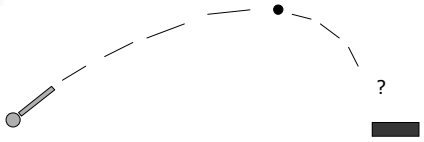
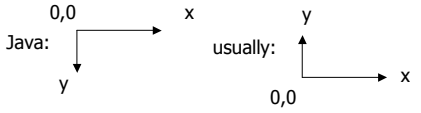
A cannon game



Simple version

- angle from command line, one shot only
- Coordinate system is "upside-down":



- Use dy(int) method to transform y coordinate:
  - public int dy(int y) { return FrameHeight-y;}

Gravity

- public class CannonBall extends Ball {

```
public CannonBall(int sx, int sy, int r,
                  double dx, double dy) {
    super(sx,sy,sr);
    setMotion(dx,dy);
    }
public static final double GravityEffect = 0.3;
public void move() {
    dy = dy + GravityEffect;
    super.move();     }}
```

- Can invoke overridden superclass method using super.methodname (cf. constructor)

Integers versus ints

- Values of primitive data types (ints, double, …) are not instances of a class
- Wrapper classes provide useful functionality: Integer, Double, …
- Both regular and static methods:

```
Integer i1 = Integer.valueOf(args[0]);
int i2 = i1.intValue();
```

Version 2: User interaction

- Button: start/terminate, turn on/off something
- Slider: one way of selecting a quantity
- Need to import both java.awt.* as well as java.awt.event.*

Inner classes

- Can define a class inside another class:

```
public class CannonWorld extends Frame {
…
   private class FireButtonListener implements
     ActionListener {..} ….}
```

- Inner classes may access all their surrounding context, including private data members and methods (see ScrollBarListener modifying private data members angle and message)

Interfaces

- An interface specifies method headers only (and maybe static data members)

```
interface ActionListener {
    public void actionPerformed(ActionEvent); }
```

- If a class implement an interface:
  - Class FireButtonListener implements ActionListener { ..}

  then it MUST provide definitions for all methods of that interface!
- A class can implement multiple interfaces (but it can only extend one class)

The Java event model

- Event: action like "click button", …
- Program response: perform some action
- "event-driven" interface
- Java: "listener" object waits for events

```
Button fire = new Button("Fire");
fire.addActionListener(new FireButtonListener);
```

- On button-press: call actionPerformed method, we MUST implement this method

Window layout

- Layout manager: controls the layout of components in a window, default is BorderLayout, can specify:
  - "North" = top
  - "South" = bottom
  - "East" = right
  - "West" = left
- E.g.: add("East",slider);

OO Progam = team of classes

- CannonWorld extends Frame
- CannonBall extends Ball
- Integer: utility, e.g. String-to-int conversion
- GUI, event classes:
  - Button, ScrollBar
  - FireButtonListener, ScrollBarListener
  - ActionEvent, AdjustmentEvent
  - BorderLayout

ALL WITHIN THREE PAGES OF CODE

Summary

- Integer: useful wrapper class
- CannonBall: inherits from Ball
- Pseudo-variable super: access both
  - Constructors of the superclass
  - (overridden) Methods of the superclass
- Inner classes: full access to surrounding context
- Interface: promise implement methods
- Event model: listeners
- GUI utilities: Button,Slider,BorderLayout