

Applets and the Graphics class

CSC 2014 – Java Bootcamp

Dr. Mary-Angela Papalaskari

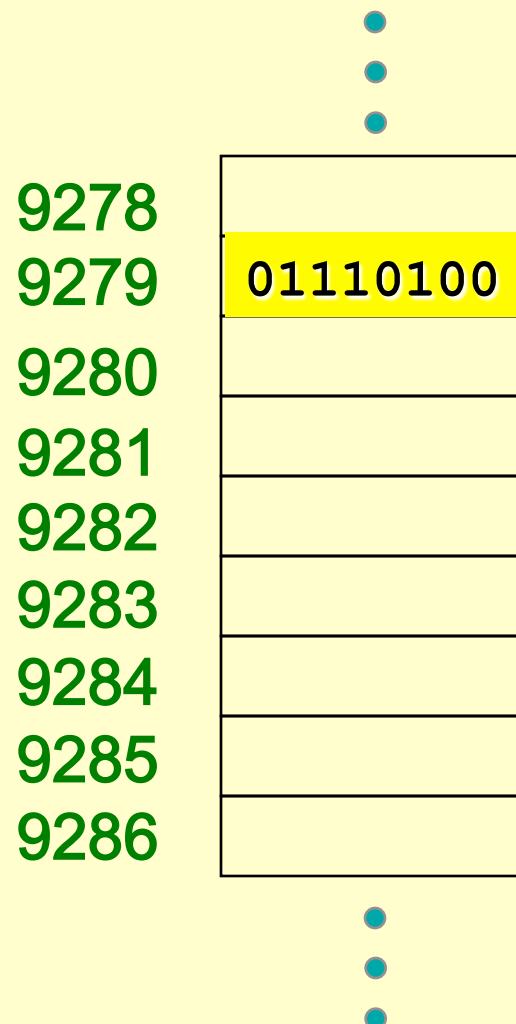
Department of Computing Sciences

Villanova University

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

Data Representation

- Computers store all information **digitally**, using **binary** codes:
 - numbers
 - text
 - images
 - audio
 - video
 - program instructions



A *byte* is a group of eight bits

01110100



- a number?
- a letter?
- the red component of a pixel?
- a program instruction?

Computing devices store & use binary codes to represent data of *all kinds*

Binary codes

<u>1 bit</u>	<u>2 bits</u>	<u>3 bits</u>	<u>4 bits</u>
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

Each additional bit doubles the number of possible codes

Binary Codes

How many codes?

- 1 bit ?**
- 2 bits ?**
- 3 bits ?**
- 4 bits ?**
- 5 bits ?**

- How many codes of N bits?
- How many bits are needed to represent 64 items?
- How many bits are needed to represent 80 items?
- How many bits are needed to represent each of the 50 states (so that each state corresponds to a unique code)?

Storage requirements examples

- If a code requires 5 bits, a document consisting of 4000 such codes will require a total of:
 $5 \times 4000 = 20,000$ bits
 - how many *bytes* is that? _____
- If a code requires 8 bits (i.e., a byte), a document consisting of 2000 such codes will require 2000 bytes.
 - how many *bits* is that? _____
- If a code requires 32 bits, a program that needs to store 2000 such codes will require
_____ bits or _____ bytes.

Storage Capacity

- Every memory device has a *storage capacity*, indicating the number of bytes it can hold

Unit	Symbol	Number of Bytes
kilobyte	KB	$2^{10} = 1024$
megabyte	MB	2^{20} (over one million)
gigabyte	GB	2^{30} (over one billion)
terabyte	TB	2^{40} (over one trillion)
petabyte	PB	2^{50} (a whole bunch)

Overview

- Binary representation
- **Data types revisited**
- Type conversions
- Image representation
- Java Applets

Variables

- A *variable* is a name for a location in memory
- A variable must be *declared* by specifying the variable's name and the type of information that it will hold

The diagram shows three variable declarations within a blue-bordered box. Two red arrows point from green labels to specific parts of the code: one arrow points from the label "data type" to the word "int" in the first declaration, and another arrow points from the label "variable name" to the word "sum" in the same declaration. The code consists of three lines:
int sum;
double milesPerGallon;
String name, petName;

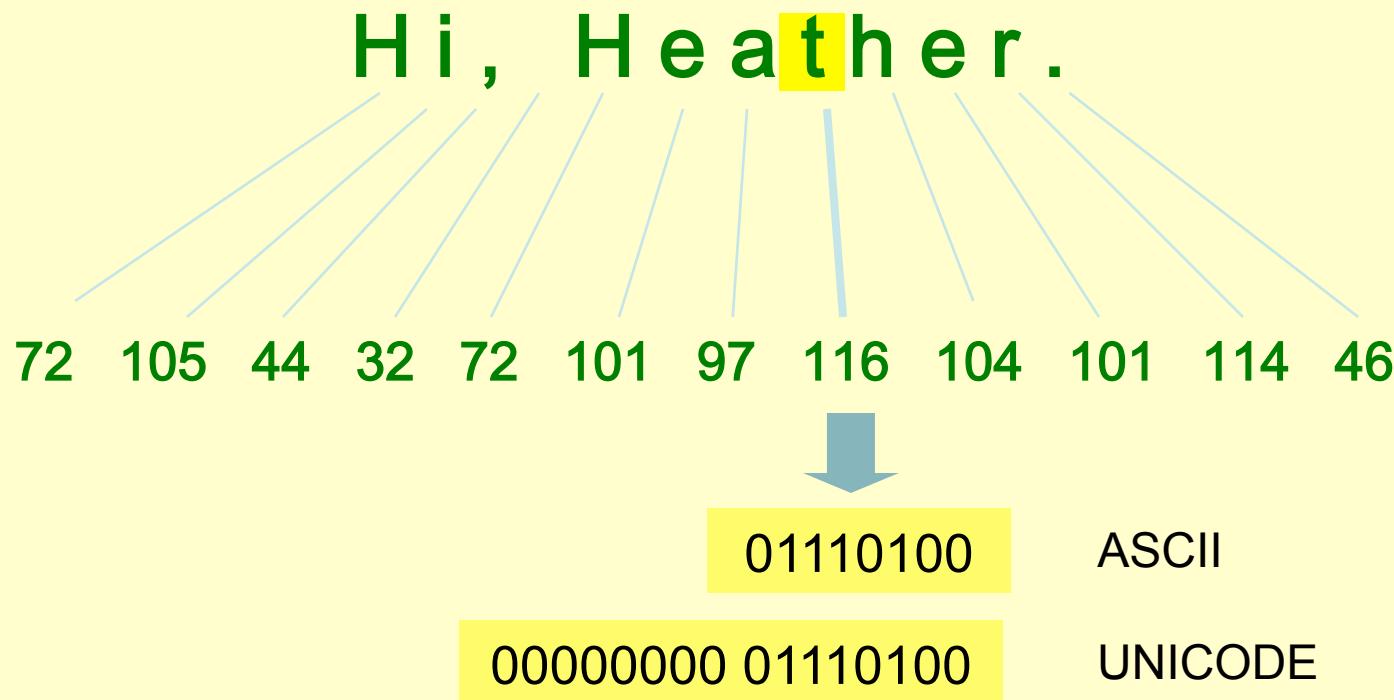
Numeric Primitive Data

- The difference between the numeric primitive types is their size and the values they can store:

<u>Type</u>	<u>Storage</u>	<u>Min Value</u>	<u>Max Value</u>
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	< -9 x 10 ¹⁸	> 9 x 10 ¹⁸
float	32 bits	+/- 3.4 x 10 ³⁸ with 7 significant digits	
double	64 bits	+/- 1.7 x 10 ³⁰⁸ with 15 significant digits	

Example: Representing Text

- Characters, including spaces, digits, and punctuation are represented by numeric codes



The **ASCII** (American Standard Code for Information Interchange) character set uses eight bits per character, allowing for 256 unique characters

The **Unicode** character set extends ASCII to sixteen bits per character, allowing for 65,536 unique characters.

Characters in Java

- A `char` variable stores a single character
- Character literals are delimited by single quotes:

'a' 'X' '7' '\$' ',' '\n'

```
char topGrade = 'A';
```

```
char terminator = ';' , separator = ' ';
```

```
char nextToTopGrade = (char)(topGrade + 1);
```

Note the difference between a primitive character variable, which holds only one character, and a `String` object, which can hold multiple characters

Overview

- Binary representation
- Data types revisited
- **Type conversions**
- Image representation
- Java Applets

Automatic type conversion

Values of different types can be combined in an assignment or an expression

- *Example:*

```
int dollars = 5;  
  
double money = dollars + 2.50;  
  
System.out.println(dollars + " dollars");
```

- These are all examples of **widening conversions**, i.e., “smaller” data type → “larger” data type

Converting from one type to another

- ***Widening conversions***

- “small” data type → “larger” one
 - eg: int → double
32 bits → 64 bits

- ***Narrowing conversions***

- “large” data type → “smaller” one

- eg: double → int

- 64 bits → 32 bits

- ***narrowing conversions can lose information!***
- ***narrowing conversions cannot happen automatically (for example, through assignment)***

Casting

- Casting forces a change of type, even if information is lost
- Can be used for both widening and narrowing conversion
- To cast, put the type in parentheses in front of the value to be converted:

```
int total = 5;
```



```
double result = (double) total / 2;
```

```
int answer = (int) result + 4;
```

```
double angle = 0; // 0 radians
```

```
int x = (int) (Math.cos(angle) * 300);
```

(cast has higher precedence
than arithmetic operators)

Data Conversion

Widening Conversions

From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

Narrowing Conversions

From	To
byte	char
short	byte or char
char	byte or short
int	byte, short, or char
long	byte, short, char, or int
float	byte, short, char, int, or long
double	byte, short, char, int, long, or float

How to use cast?

Forcing floating point division between `int` expressions

```
int qp = 35;  
  
int credits = 10;  
  
double gpa = (double) qp / credits;
```

gpa should be 3.5

?

```
int qp = 35;  
  
int credits = 10;  
  
double gpa = (double) (qp / credits);
```

How to use cast?

*Scaling a **double** and converting to **int***

```
double gpa = 3.2;  
int gpaPercent = (int) (gpa / 4) * 100;
```

gpaPercent should be 80

?

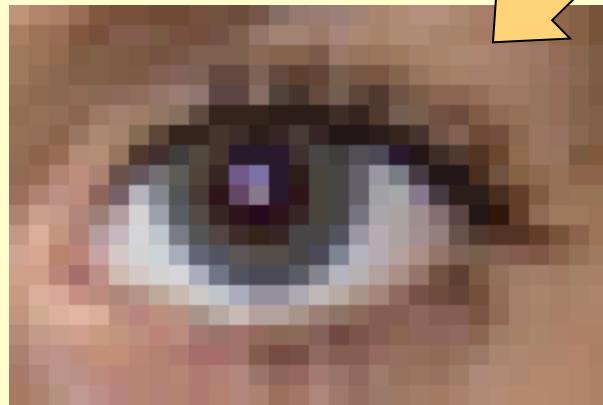
```
double gpa = 3.2;  
int gpaPercent = (int) ((gpa / 4) * 100);
```

Overview

- Binary representation
- Data types revisited
- Type conversions
- **Image representation**
- Java Applets

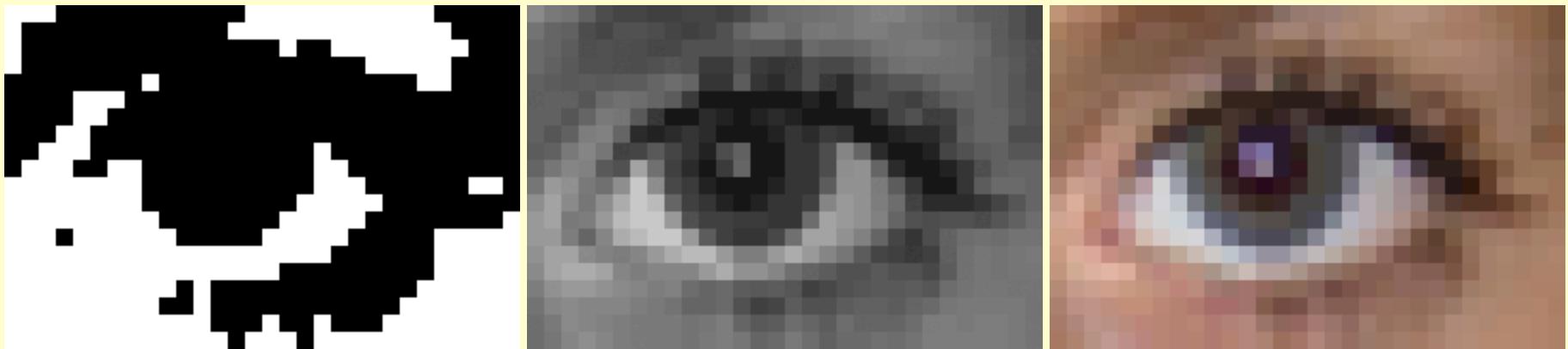
What's a picture?

- programs represent pictures as grids of picture elements or ***pixels***



Stephanos with his eraser collection

Representing Images



Bitmap

1 bit

Grayscale

8 bits

RGB Color

3 colors: red, green, blue

8 bits/color

24 bits

Example: Representing Pixels



fff

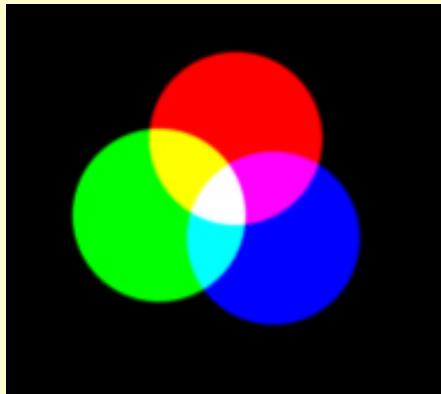
Additive/Subtractive Color

We choose 3 primary colors that can be combined to produce almost all visible colors

Additive primaries

- combining light

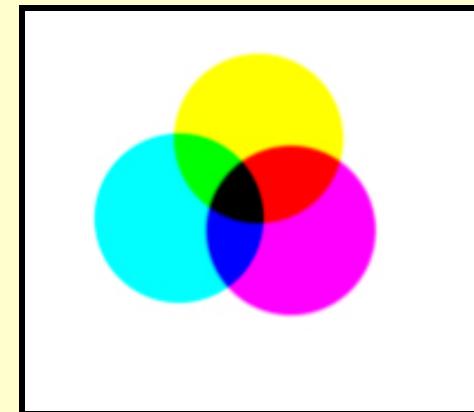
Red Green Blue



Subtractive primaries

- combining ink, thus subtracting light

Cyan Yellow Magenta



Bitmap image



A 2000 by 1000 bitmap image contains

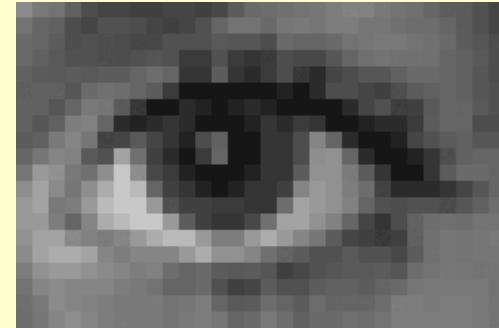
Bitmap
1 bit/pixel

_____ pixels.

Thus, we need _____ bits to

to represent this image, or _____ bytes

Grayscale Image



A 2000 by 1000 grayscale image contains

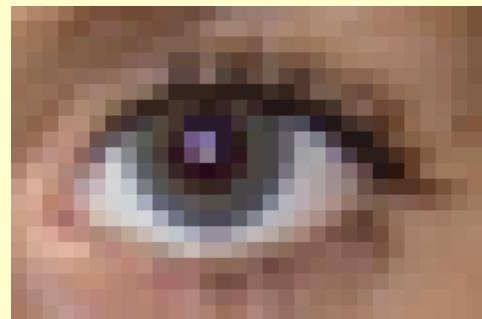
Grayscale
8 bits/pixel

_____ pixels.

Thus, we need _____ bytes

to represent this image, or _____ bits

RGB Image



A 2000 by 1000 RGB image contains
_____ pixels.

RGB color
24 bits/pixel
3 bytes/pixel

Thus, we need _____ bytes

to represent this image, or _____ bits

Graphics and images in Java

- Images are represented as objects in Java.
- Color components and positions for pixels can be represented as integers (but also in other ways).
- Java libraries allow flexibility, providing many alternative ways of representing and processing images.
- We will be using the *Graphics* class of the *awt* package and *Japplet* from the *swing*

Overview

- Binary representation
- Data types revisited
- Type conversions
- Image representation
- **Java Applets**

Applets

- A Java *applet* is a program that is intended to be transported over the Web and executed using a web browser
- An applet doesn't have a `main` method
 - The type of program we have seen so far is a Java *application* - a stand-alone program with a `main` method

Example: Einstein.java

```
*****  
// Einstein.java          Author: Lewis/Loftus  
//  
// Demonstrates a basic applet.  
*****  
  
import javax.swing.JApplet;  
import java.awt.*;  
  
public class Einstein extends JApplet  
{  
    //-----  
    // Draws a quotation by Albert Einstein among some shapes.  
    //-----  
    public void paint (Graphics page)  
    {  
        page.drawRect (50, 50, 40, 40);      // square  
        page.drawRect (60, 80, 225, 30);    // rectangle  
        page.drawOval (75, 65, 20, 20);    // circle  
        page.drawLine (35, 60, 100, 120);  // line  
  
        page.drawString ("Out of clutter, find simplicity.", 110, 70);  
        page.drawString ("-- Albert Einstein", 130, 100);  
    }  
}
```

```

//*****
// Einstein.java          Author: Lewis/Lof
//
// Demonstrates a basic applet.
//*****

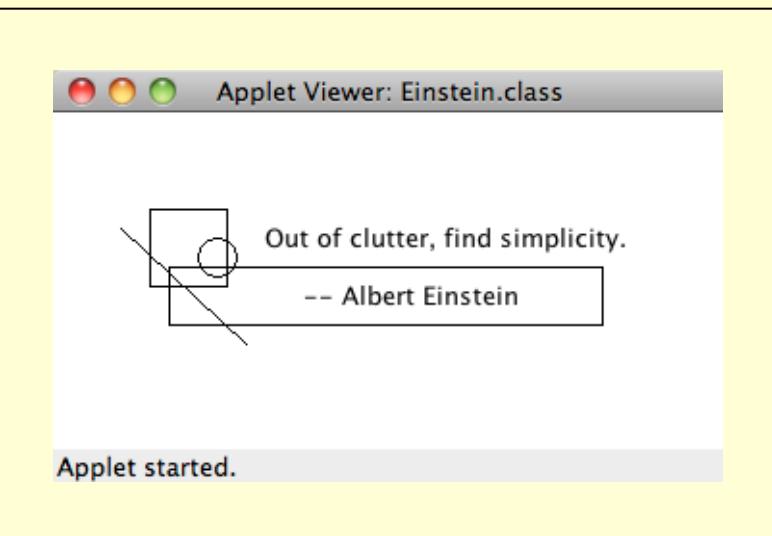

import javax.swing.JApplet;
import java.awt.*;


public class Einstein extends JApplet
{
    //-----
    // Draws a quotation by Albert Einstein among some shapes.
    //-----

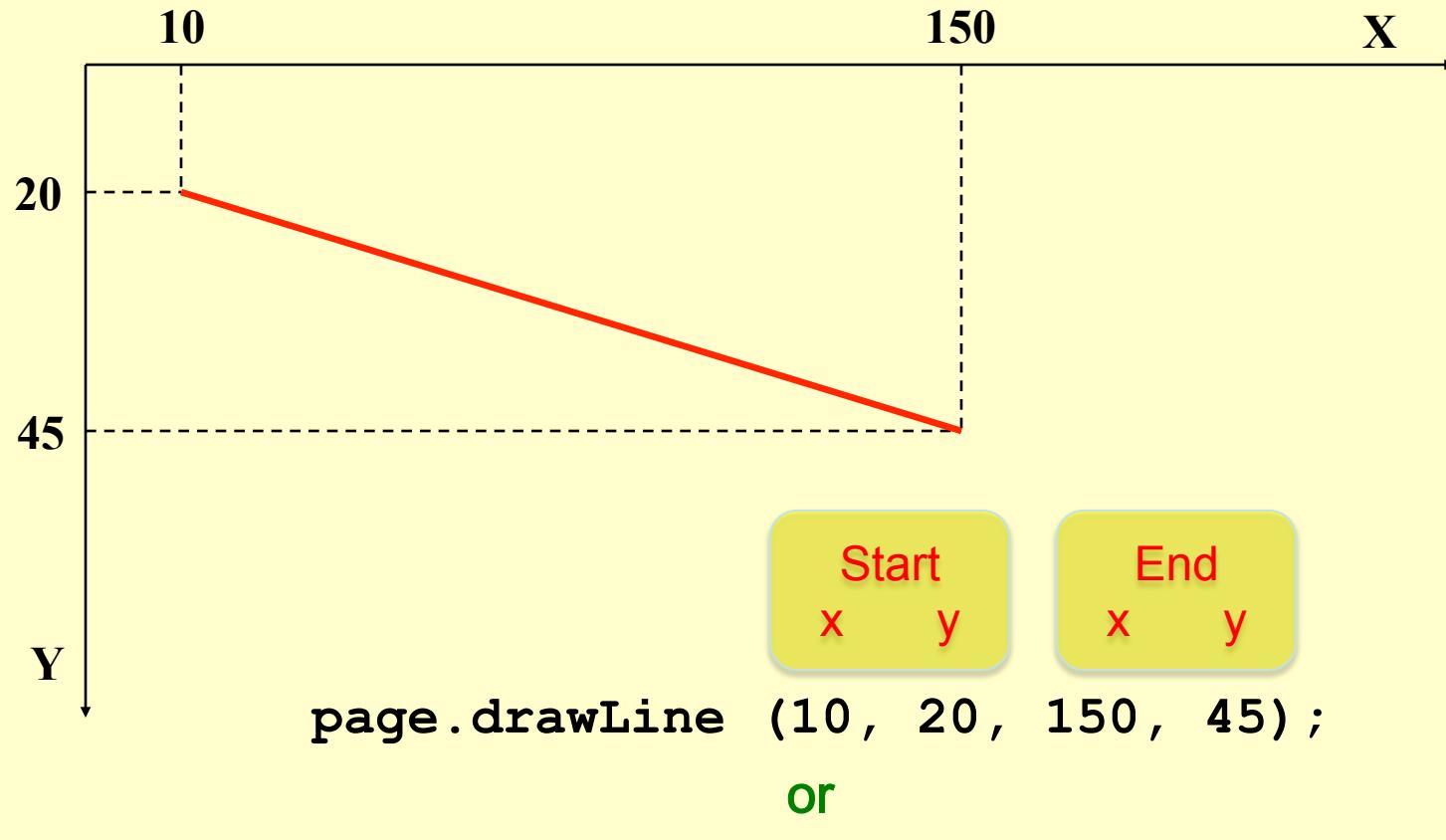

    public void paint (Graphics page)
    {
        page.drawRect (50, 50, 40, 40);      // square
        page.drawRect (60, 80, 225, 30);    // rectangle
        page.drawOval (75, 65, 20, 20);    // circle
        page.drawLine (35, 60, 100, 120);   // line

        page.drawString ("Out of clutter, find simplicity.", 110, 70);
        page.drawString ("-- Albert Einstein", 130, 100);
    }
}

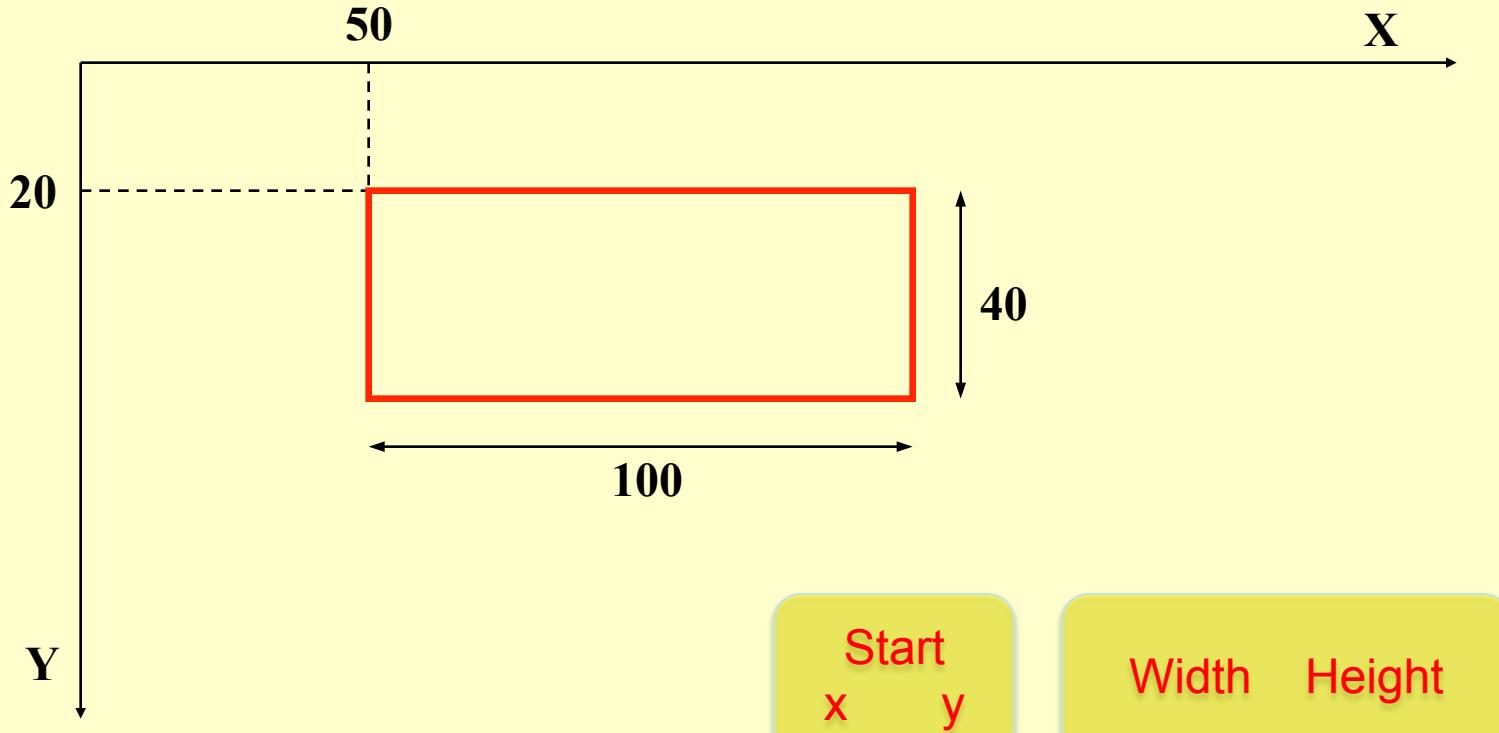
```



Drawing a Line

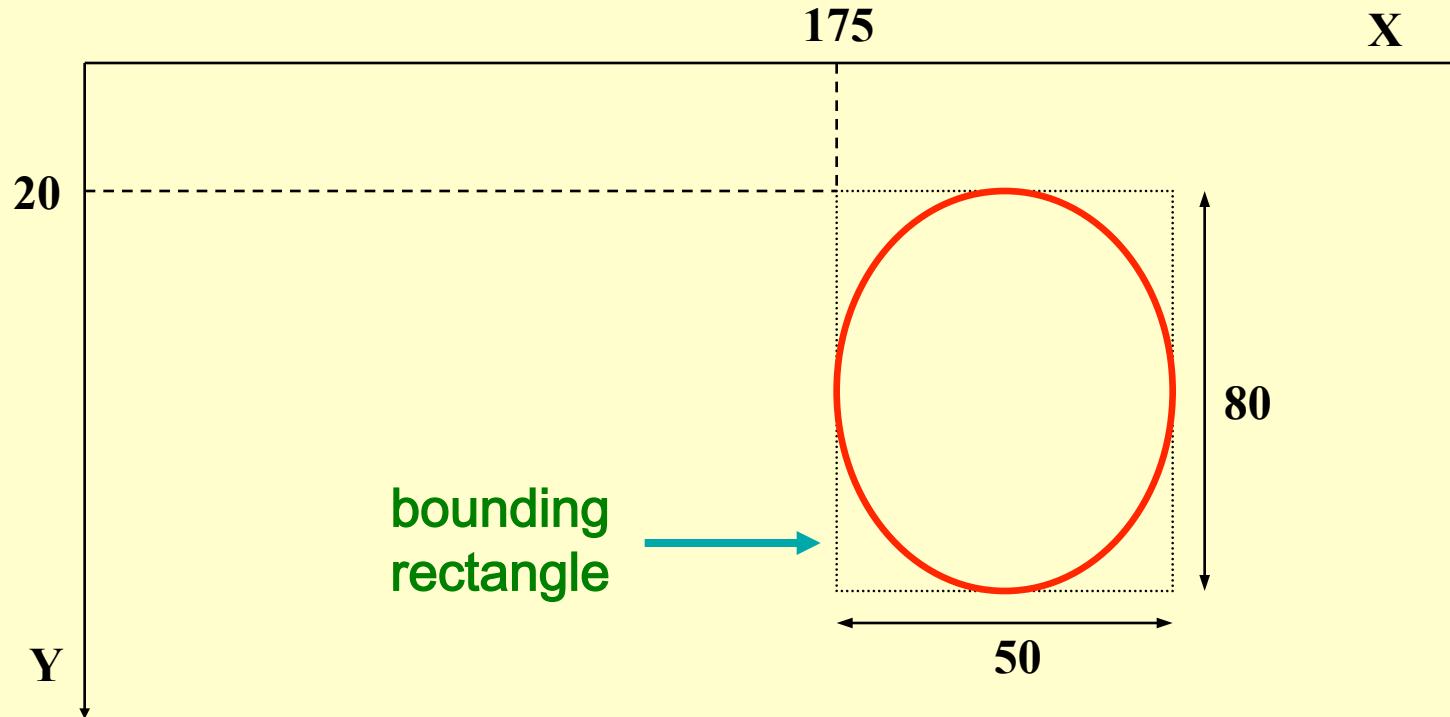


Drawing a Rectangle



```
page.drawRect (50, 20, 100, 40);
```

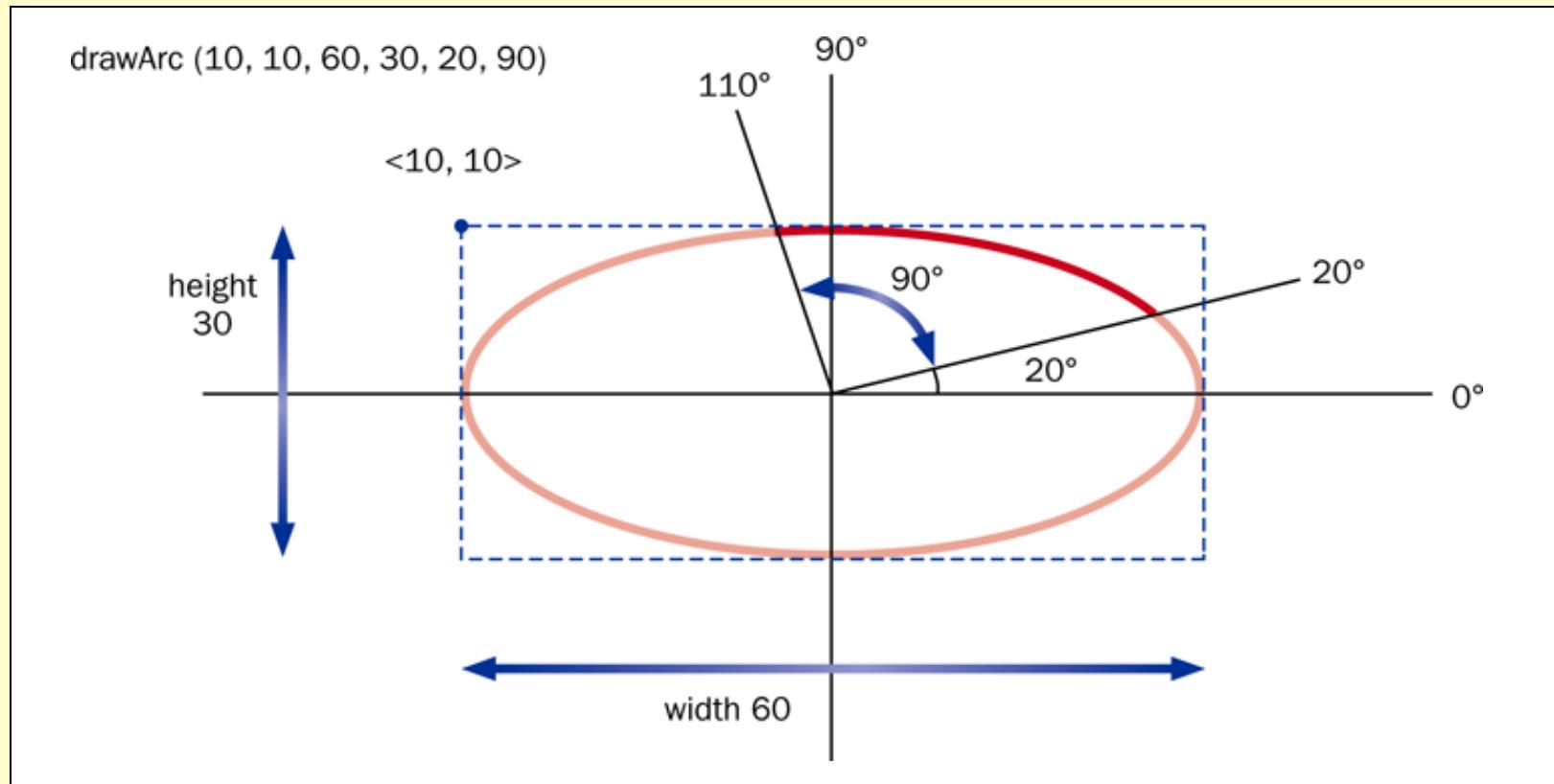
Drawing an Oval



```
page.drawOval (175, 20, 50, 80);
```

Drawing an Arc

- An arc is defined by an oval, a start angle, and an arc angle:

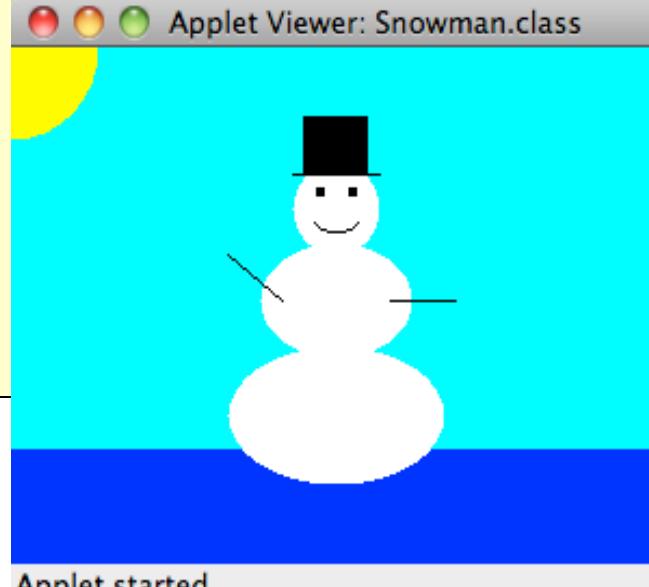


Filled vs unfilled shapes

- Instead of using drawRect(), drawOval() etc, we can use fillRect(), fillOval() etc
- We can set the color using setColor()
- [See Snowman.java](#)
- [See also Snowman applet on a webpage](#)

```
//*****  
//  Snowman.java          Author: Lewis/Loftus  
//  
//  Demonstrates basic drawing methods and the use of color.  
//*****  
  
import javax.swing.JApplet;  
import java.awt.*;  
  
public class Snowman extends JApplet  
{  
    //-----  
    //  Draws a snowman.  
    //-----  
    public void paint (Graphics page)  
    {  
        final int MID = 150;  
        final int TOP = 50;  
  
        setBackground (Color.cyan);  
  
        page.setColor (Color.blue);  
        page.fillRect (0, 175, 300, 50); // ground  
  
        page.setColor (Color.yellow);  
        page.fillOval (-40, -40, 80, 80); // sun
```

continued



continued

```
page.setColor (Color.white);
page.fillOval (MID-20, TOP, 40, 40);
page.fillOval (MID-35, TOP+35, 70, 50); // upper torso
page.fillOval (MID-50, TOP+80, 100, 60); // lower torso

page.setColor (Color.black);
page.fillOval (MID-10, TOP+10, 5, 5); // left eye
page.fillOval (MID+5, TOP+10, 5, 5); // right eye

page.drawArc (MID-10, TOP+20, 20, 10, 190, 160); // smile

page.drawLine (MID-25, TOP+60, MID-50, TOP+40); // left arm
page.drawLine (MID+25, TOP+60, MID+55, TOP+60); // right arm

page.drawLine (MID-20, TOP+5, MID+20, TOP+5); // brim of hat
page.fillRect (MID-15, TOP-20, 30, 25); // top of hat
}

}
```

Encoding RGB

- Each component color (red, green, and blue) is encoded as a single byte
- Colors go from (0,0,0) to (255,255,255)
 - (0,0,0) = black
 - (255,255,255) = white
 - If all three components are the same, the color is in greyscale
 - eg: (50,50,50)

	1	2	3
1	100,10,5	5,10,100	255,0,0
2	0,0,0	50,50,50	0,100,0

The Java Color Class

- A color in a Java program is represented as an object created from the `Color` class
- The `Color` class also contains several predefined colors, eg:

<u>Object</u>	<u>RGB Value</u>
<code>Color.black</code>	0, 0, 0
<code>Color.blue</code>	0, 0, 255
<code>Color.cyan</code>	0, 255, 255
<code>Color.orange</code>	255, 200, 0
<code>Color.white</code>	255, 255, 255
<code>Color.yellow</code>	255, 255, 0

- Using a color: `page.setColor(Color.blue);`
- Creating a new color:

```
Color salmon = new Color(255, 140, 128);
page.setColor(salmon);
```

Translation of programs into machine code

High-level language

```
public class Einstein extends JApplet
{
    //-
    // Draws a quotation by Albert Einstein
    //
    public void paint (Graphics page)
    {
        page.drawRect (50, 100, 400, 200);
        page.drawRect (50, 200, 400, 300);
        page.drawOval (150, 150, 300, 250);
        page.drawLine (150, 150, 250, 250);

        page.drawString ("Out of clutter, find
        page.drawString ("-- Albert Einstein
    }
}
```

source
code

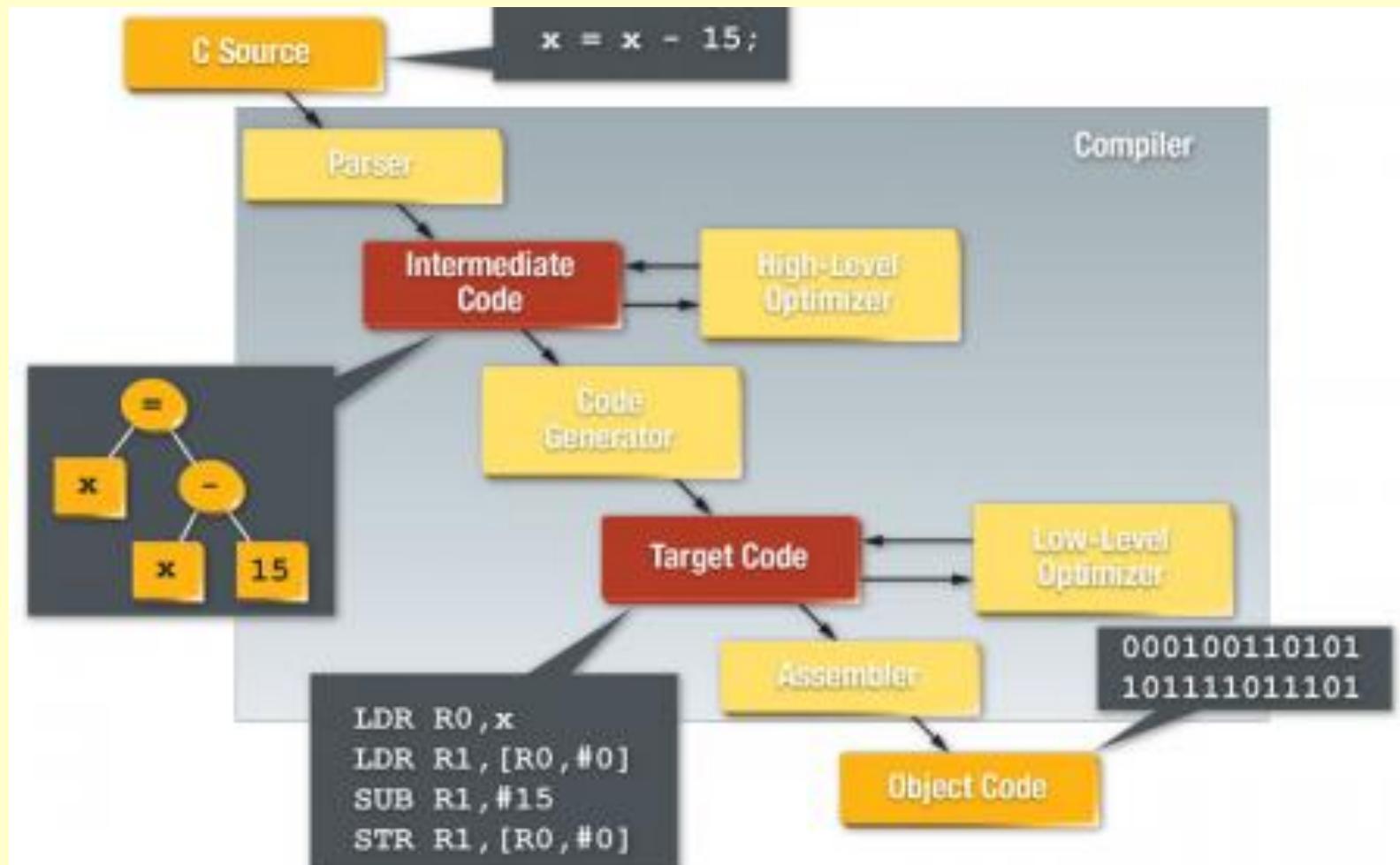
compiler

Machine
code

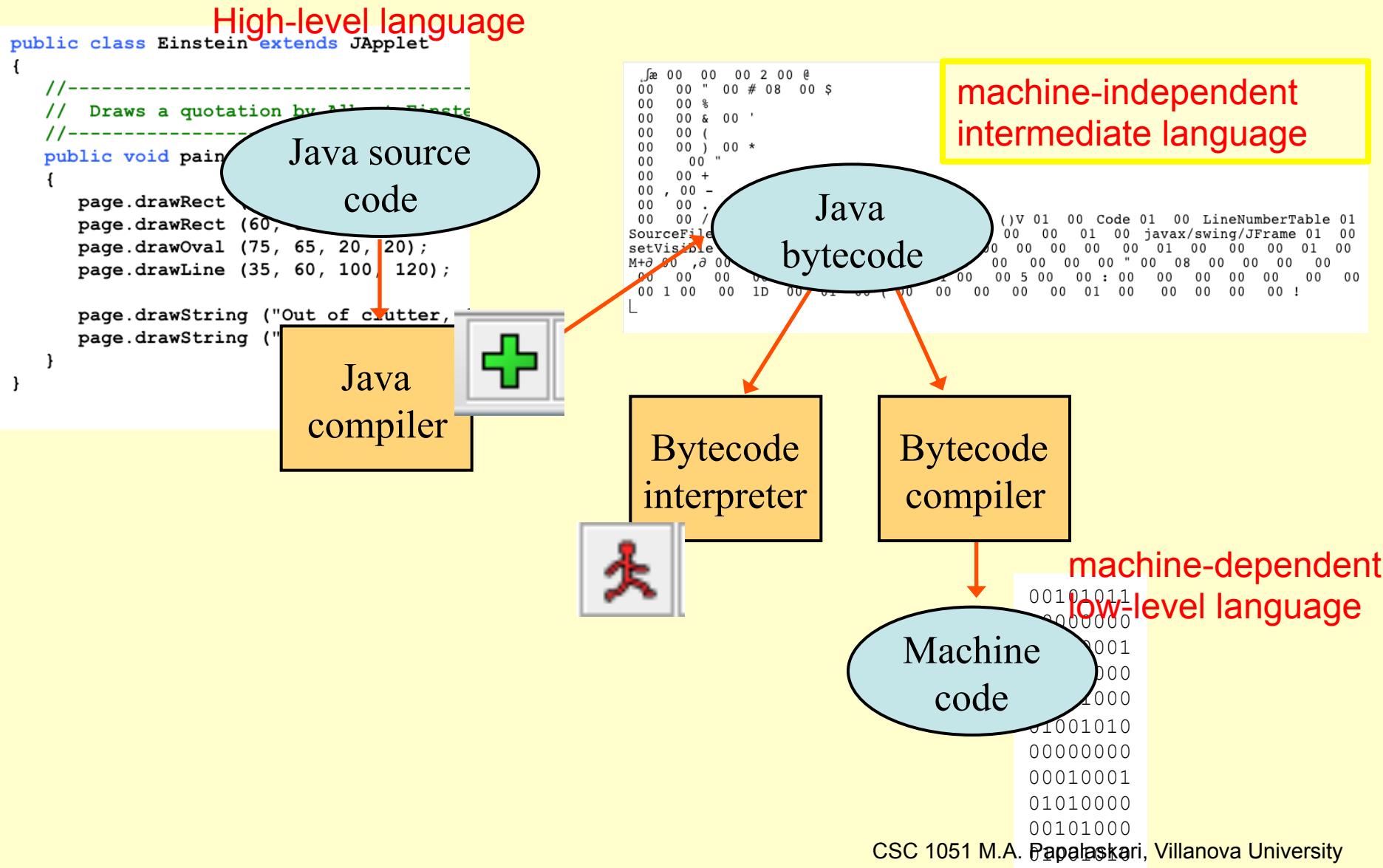
machine-dependent
low-level language

```
00101011
00000000
01
1010
00000000
00010001
01010000
00101000
01001010
```

Translation of programs into machine code
- reality is usually a bit more complicated



Java Translation is different



The HTML applet Tag

- An applet is embedded into an HTML file using a tag that references the bytecode file of the applet
- The bytecode version of the program is transported across the web and executed by a Java interpreter that is part of the browser.

```
<html>
  <head>
    <title>The Einstein Applet</title>
  </head>
  <body>
    <applet code="Einstein.class" width=350 height=175>
    </applet>
  </body>
</html>
```