Data Representation and Applets

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
http://www.csc.villanova.edu/~map/1051/
Overview

• Binary representation
  • Image representation
  • Java Applets
  • Data types revisited
Data Representation

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

CSC 1051 M.A. Papalaskari, Villanova University
Why Binary Numbers?

• Simplest way to represent digital information:
  – **Electronic circuits**: high/low voltage
  – **Magnetic devices** (eg hard drive): positive/negative
  – **Optical devices** (eg DVD): light reflected/not reflected due to microscopic grooves

A binary digit is called a **bit** - *binary digit*
A *byte* is a group of eight bits

- 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

<table>
<thead>
<tr>
<th>1 bit</th>
<th>2 bits</th>
<th>3 bits</th>
<th>4 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>010</td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>011</td>
<td>0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
<td>0101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
<td>0110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>0111</td>
</tr>
</tbody>
</table>

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:
  \[ 5 \times 4000 = 20,000 \text{ bits} \]
  – how many \textit{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 \textit{bits} is that? _______________

• If a code requires 32 bits, a program that needs to store 2000 such codes will require
  \[ \underline{_______} \text{ bits} \quad \text{or} \quad \underline{_______} \text{ bytes} \]
Storage Capacity

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

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Number of Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>kilobyte</td>
<td>KB</td>
<td>$2^{10} = 1024$</td>
</tr>
<tr>
<td>megabyte</td>
<td>MB</td>
<td>$2^{20}$ (over one million)</td>
</tr>
<tr>
<td>gigabyte</td>
<td>GB</td>
<td>$2^{30}$ (over one billion)</td>
</tr>
<tr>
<td>terabyte</td>
<td>TB</td>
<td>$2^{40}$ (over one trillion)</td>
</tr>
<tr>
<td>petabyte</td>
<td>PB</td>
<td>$2^{50}$ (a whole bunch)</td>
</tr>
</tbody>
</table>
Overview

- Binary representation
- **Image representation**
- Java Applets
- Data types revisited
What’s a picture?

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

Bitmap
1 bit

Grayscale
8 bits

RGB Color
3 colors: red, green, blue
8 bits/color
24 bits
Example: Representing Pixels

\[
\text{Color (} \begin{array}{ccc}
01110100 \\
01010110 \\
10001110 \\
\end{array} \text{)}
\]

red=116, green=86, blue=142

\[
x = 11, \quad y = 8
\]
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

___________ pixels.

Thus, we need ____________ bits to

to represent this image, or ____________ bytes
Grayscale Image

A 2000 by 1000 grayscale image contains ___________ pixels.

Thus, we need ___________ bytes to represent this image, or ___________ bits.
RGB Image

A 2000 by 1000 RGB image contains ____________ pixels.

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
- Image representation
- **Java Applets**
- Data types revisited
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

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);
    }
}

CSC 1051 M.A. Papalaskari, Villanova University
import javax.swing.JApplet;
import java.awt.*;

public class Einstein extends JApplet{
    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); // line
        page.drawString("-- Albert Einstein", 130, 100); // line
    }
}
Drawing a Line

```
page.drawLine (10, 20, 150, 45);
page.drawLine (150, 45, 10, 20);
```

Start

```
  x   y
```

End

```
  x   y
```

```
page.drawLine (10, 20, 150, 45);
  or
page.drawLine (150, 45, 10, 20);
```
Drawing a Rectangle

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

CSC 1051 M.A. Papalaskari, Villanova University
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
import javax.swing.JApplet;
import java.awt.*;

public class Snowman extends JApplet {

    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

}
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:

<table>
<thead>
<tr>
<th>Object</th>
<th>RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color.black</td>
<td>0, 0, 0</td>
</tr>
<tr>
<td>Color.blue</td>
<td>0, 0, 255</td>
</tr>
<tr>
<td>Color.cyan</td>
<td>0, 255, 255</td>
</tr>
<tr>
<td>Color.orange</td>
<td>255, 200, 0</td>
</tr>
<tr>
<td>Color.white</td>
<td>255, 255, 255</td>
</tr>
<tr>
<td>Color.yellow</td>
<td>255, 255, 0</td>
</tr>
</tbody>
</table>

• Using a color:  

```java
page.setColor(Color.blue);
```

• Creating a new color:

```java
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, 50, 100, 100);
        page.drawRect (150, 150, 200, 200);
        page.drawOval (250, 250, 300, 300);
        page.drawLine (350, 350, 400, 400);
        page.drawString ("Out of clutter, I
                        page.drawString ("-- Albert Einstein
    }
}
```

source code

compiler

machine-dependent
low-level language

Machine code
Java Translation is different

- High-level language
- Machine code
- Intermediate language

Java source code

```
public class Einstein extends JApplet
{
    // Draws a quotation by Albert Einstein
    public void paint()
    {
        page.drawRect (60, 70, 100, 120);
        page.drawOval (75, 65, 20, 20);
        page.drawLine (35, 60, 100, 120);
        page.drawString ("Out of clutter, page.drawString ("
    }
```

Bytecode compiler

Bytecode interpreter

CSC 1051 M.A. Papalaskari, Villanova University
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
<html>
  <head>
    <title>The Einstein Applet</title>
  </head>
  <body>
    <applet code="Einstein.class" width=350 height=175>
    </applet>
  </body>
</html>
```
Overview

• Binary representation
• Image representation
• Java Applets
• Data types revisited
Numeric Primitive Data

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 x 10^{18}</td>
<td>&gt; 9 x 10^{18}</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^{38} with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 10^{308} with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>

CSC 1051 M.A. Papalaskari, Villanova University
Automatic type conversion

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

• Example:

```java
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 \(\rightarrow\) “larger” data type
Converting from one type to another

- **Widening conversions**
  - “small” data type $\rightarrow$ “larger” one
  - eg: int $\rightarrow$ double
  - 32 bits $\rightarrow$ 64 bits

- **Narrowing conversions**
  - “large” data type $\rightarrow$ “smaller” one
  - eg: double $\rightarrow$ int
  - 64 bits $\rightarrow$ 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:

```java
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

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>short, int, long, float, or double</td>
</tr>
<tr>
<td>short</td>
<td>int, long, float, or double</td>
</tr>
<tr>
<td>char</td>
<td>int, long, float, or double</td>
</tr>
<tr>
<td>int</td>
<td>long, float, or double</td>
</tr>
<tr>
<td>long</td>
<td>float or double</td>
</tr>
<tr>
<td>float</td>
<td>double</td>
</tr>
</tbody>
</table>

### Narrowing Conversions

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>char</td>
</tr>
<tr>
<td>short</td>
<td>byte or char</td>
</tr>
<tr>
<td>char</td>
<td>byte or short</td>
</tr>
<tr>
<td>int</td>
<td>byte, short, or char</td>
</tr>
<tr>
<td>long</td>
<td>byte, short, char, or int</td>
</tr>
<tr>
<td>float</td>
<td>byte, short, char, int, or long</td>
</tr>
<tr>
<td>double</td>
<td>byte, short, char, int, long, or float</td>
</tr>
</tbody>
</table>
How to use cast?

Forcing floating point division between int expressions

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

```
int qp = 35;
int credits = 10;

double gpa = (double) (qp / credits);
```

gpa should be 3.5
How to use cast?

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

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

`gpaPercent` should be 80

```java
double gpa = 3.2;
int gpaPercent = (int) ((gpa / 4) * 100);
```
Characters in Java

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

**Hi, Heather.**

- 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.

The ASCII: 01110100

The Unicode: 00000000 01110100
Characters in Java

• A `char` variable stores a single character.

• Character literals are delimited by single quotes:

  `'a'   'X'    '7'    '$'    ','    '
`

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

```java
char grade = 'A';
char terminator = ';', separator = ' ', newline = '
';

char letter = 't';
char next = letter++; // 'u'
```
**char ↔ int Conversion**

- A `char` variable is stored as its unicode representation
- `char ↔ int` conversion: convert between char and its unicode.
- e.g.: ‘t’ ⇔ 116
- increment and decrement of `char` variables takes you up and down in alphabetical order (codes are in numeric sequence)

```java
char letter = 't';
int letterCode = letter;    // 116 (code for 't')

char next = letter++;       // 'u'
int nextCode = next;        // 117 (code for 'u')

char nextAlt = (char) nextCode; // 'u'
                          // narrowing conversion - requires cast
char whatsthis = (char) ('A' + 3);    // ???
int num = (int) (letter - 'p');      // ???
```