Applets and the Graphics class

CSC 2014 – Java Bootcamp

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

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

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>10</td>
<td>010</td>
<td>011</td>
<td>0010</td>
</tr>
<tr>
<td>11</td>
<td>011</td>
<td>100</td>
<td>0011</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>110</td>
<td>0100</td>
</tr>
<tr>
<td>101</td>
<td>110</td>
<td>111</td>
<td>0101</td>
</tr>
<tr>
<td>1000</td>
<td>1010</td>
<td>1011</td>
<td>1010</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1100</td>
</tr>
<tr>
<td>1110</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
</tr>
</tbody>
</table>

Each additional bit doubles the number of possible codes
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>

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.

```
int sum;
double milesPerGallon;
String name, petName;
```

Overview

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

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; $-9 \times 10^{18}$</td>
<td>&gt; $9 \times 10^{18}$</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- $3.4 \times 10^{38}$ with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- $1.7 \times 10^{308}$ with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
Example: Representing Text

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

![Example: Representing Text](image)

```plaintext
01110100
```

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'

```plaintext
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)

```java
int total = 5;
double result = (double) total / 2;
int answer = (int) result + 4;
```

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;
```

How to use cast?

*Forcing floating point division between int expressions*

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

```java
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;
```

**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

**Overview**

- Binary representation
- Data types revisited
- Type conversions
- Image representation
- Java Applets
Example: Representing Pixels

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

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 package.

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

```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);
    }
}
```

Drawing a Line

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

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

Drawing an Oval

```java
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`
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, e.g.

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

```
Color salmon = new Color(255, 140, 128);
page.setColor(salmon);
```
Java Translation is different

Java Translation is different between high-level language (Java source code) and machine-dependent code (bytecode). The machine-independent intermediate language (Bytecode) is generated by the Java compiler and is executed by the Bytecode compiler, which is part of the Bytecode interpreter. The Bytecode interpreter is responsible for interpreting the bytecode and executing the program on the machine.

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>
```