Programming in Java: lecture 4

- Exceptions and try...catch
- Overview – static vs. non static
- GUI programming – Applets
- Black Boxes
  - Subroutines
  - Local and Global variables
  - Parameters – formal and actual
  - Overloading

Slides made for use with ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Some figures are taken from ”Introuction to Programming Using Java, Version 5.0” by David J. Eck
Lecture 3 covers Section 3.7 to 3.8 and 4.1 to 4.3

Exceptions

- Last time: Normal flow of control
  - Why do we need something different
  - Handle errors somewhere else then where they happen
- Exception – the exception is an Object
- try...catch statements

try...catch

- Formal syntax

```
try {
   ⟨statements-1⟩
}
catch ( ⟨exception-class-name⟩  ⟨variable-name⟩ ) {
   ⟨statements-2⟩
}
```

try...catch

- Example

```
try {
   double x;
   x = Double.parseDouble(str);
   System.out.println( "The number is " + x );
}
catch ( NumberFormatException e ) {
   System.out.println( "Not a legal number." );
}
```

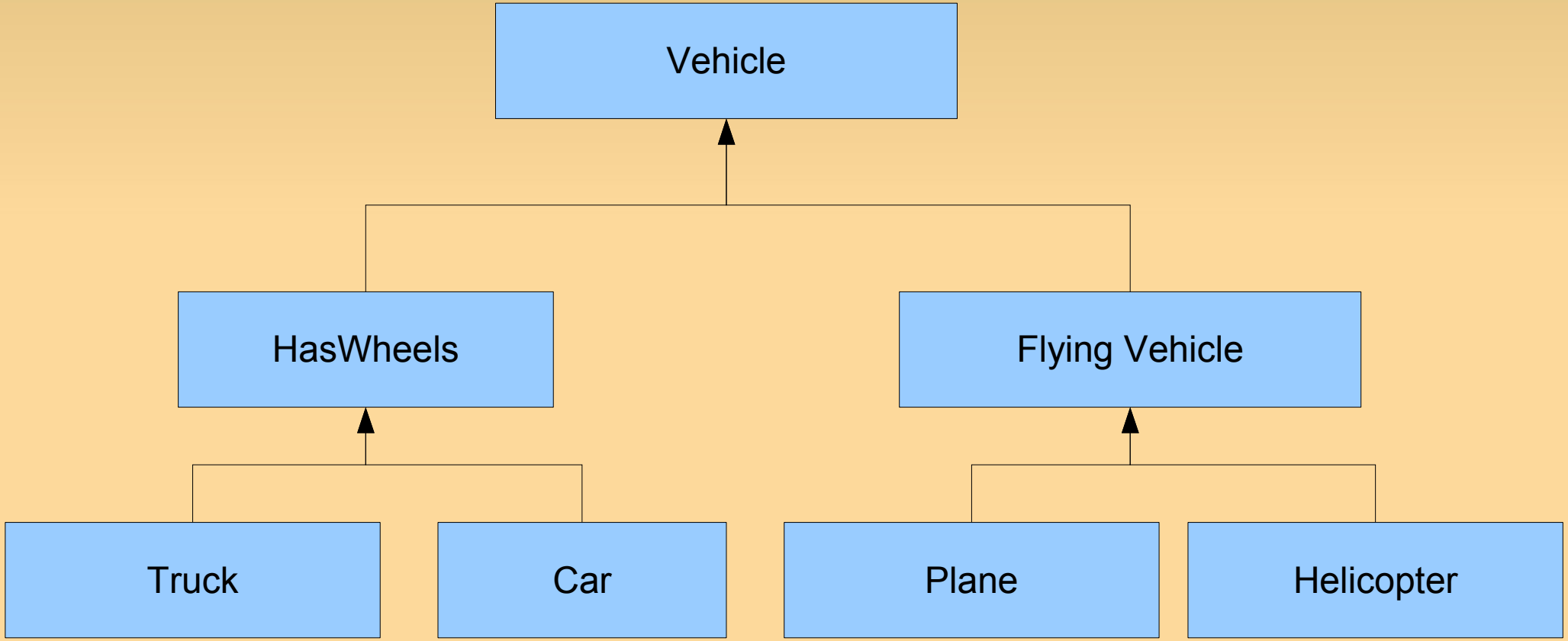
Example 2

```
Day  weekday;  // User's response as a value of type Day.
while ( true ) {
   String response;  // User's response as a String.
   TextIO.put("Please enter a day of the week: ");
   response = TextIO.getln();
   response = response.toUpperCase();
   try {
      weekday = Day.valueOf(response);
      break;
   }
   catch ( IllegalArgumentException e ) {
      TextIO.putln( response + " is not the name of a day of the week." );
   }
}
```

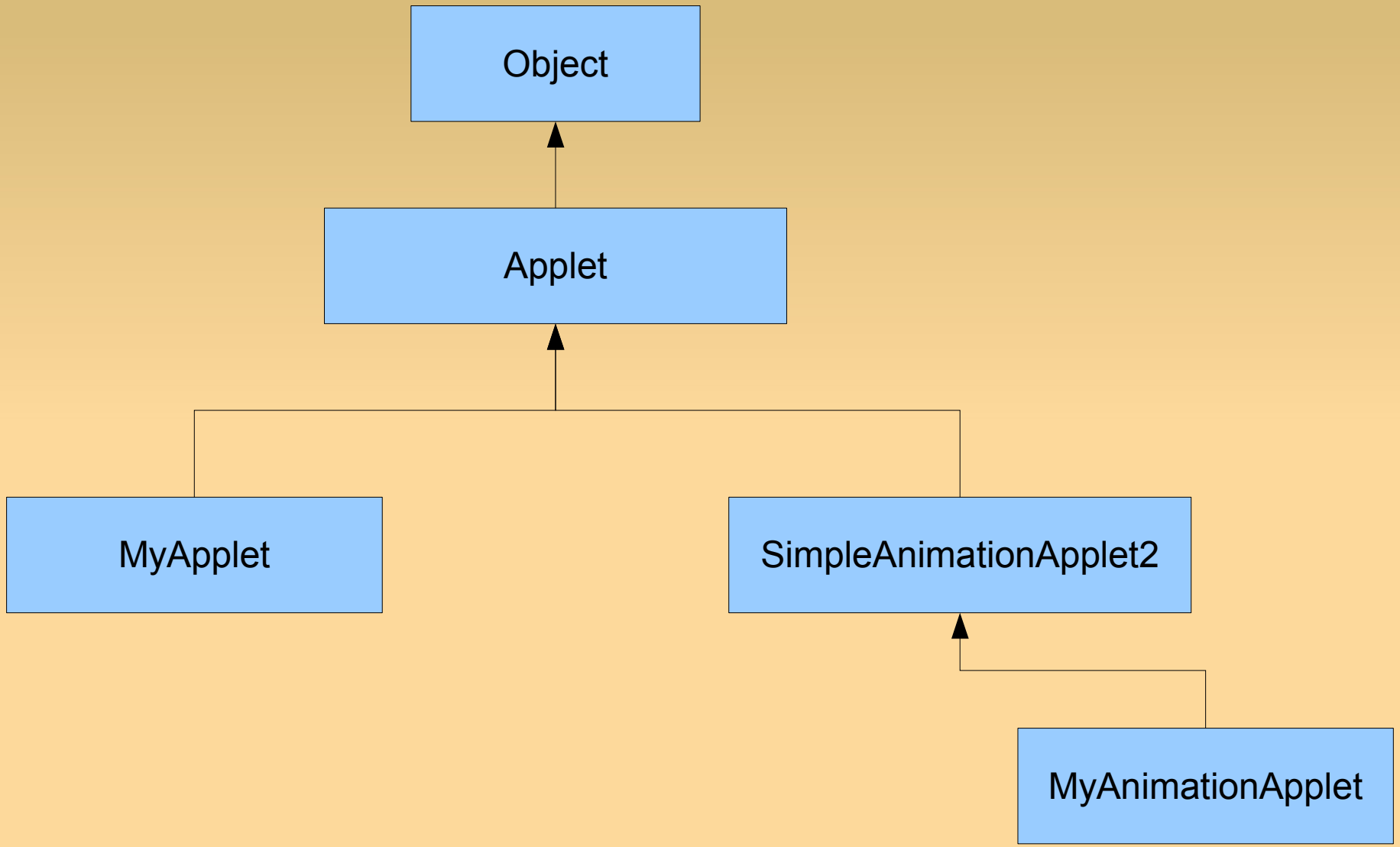
Exceptions in TextIO

- When reading user input TextIO handles error itself
- When reading from a file, this is not possible.
- Thus is throws an error that you have to catch.

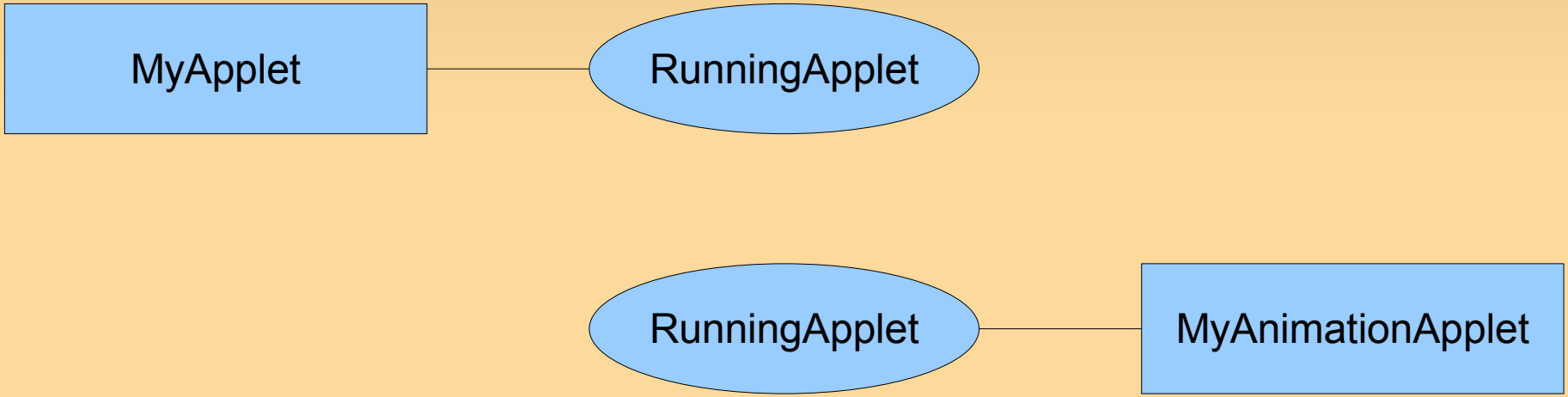
Overview


Vehicle
HasWheels
Flying Vehicle
Truck
Car
Plane
Helicopter

Classes


Object
Applet
MyApplet
SimpleAnimationApplet2
MyAnimationApplet

Static vs. non-static

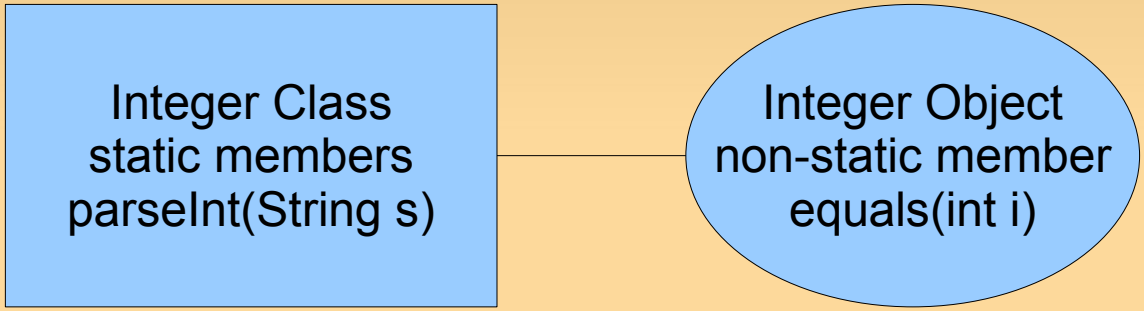


Math vs. String

- Static memebers
  - Math.rand()
  - Integer.parseInt(“45”)
- Non-static members
  - s1 = “Dette bliver et object”
  - s1.equals(“hej”)

Example: Integer

- Class vs. Object



GUI programming

- GUI = Graphical User Interface
- Applets
  - Making an applet class
  - When running the applet class an object is created
- Regular programs – as in previous lectures
  - public static void main(String[] args) {...
- Applets
  - public void paint(Graphics g) {...

Syntax

- import – packages
- extends

```
import java.awt.*;
import java.applet.*;

public class ⟨name-of-applet⟩ extends Applet {

    public void paint(Graphics g) {
       ⟨statements⟩
    }

}
```

Graphics

- java.awt vs. javax.swing
- Applet vs. Japplet
- Methods on Graphics obejct
  - g.setColor(c)
    - c is of type enum Color: ex Color.RED, Color.BLUE
  - g.drawRect(x,y,w,h)
  - g.fillRect(x,y,w,h)
  - Draws rectangles

```
import java.awt.*;
import java.applet.Applet;

public class StaticRects extends Applet {

   public void paint(Graphics g) {

         // Draw a set of nested black rectangles on a red background.
         // Each nested rectangle is separated by 15 pixels on
         // all sides from the rectangle that encloses it.

      int inset;    // Gap between borders of applet
                    //         and one of the rectangles.

      int rectWidth, rectHeight;   // The size of one of the rectangles.

      g.setColor(Color.red);
      g.fillRect(0,0,300,160);  // Fill the entire applet with red.

      g.setColor(Color.black);  // Draw the rectangles in black.

      inset = 0;

      rectWidth = 299;    // Set size of first rect to size of applet.
      rectHeight = 159;

      while (rectWidth >= 0 && rectHeight >= 0) {
         g.drawRect(inset, inset, rectWidth, rectHeight);
         inset += 15;         // Rects are 15 pixels apart.
         rectWidth -= 30;     // Width decreases by 15 pixels
                              //              on left and 15 on right.
         rectHeight -= 30;    // Height decreases by 15 pixels
                              //              on top and 15 on bottom.
      }

   }  // end paint()

}  // end class StaticRects
```

Animation

- Extend SimpleAnimationApplet2
- implement drawFrame() method
- use this.getFrameNumer() in some way

Black Boxes

- Why?
  - Hiding details and complexity
- Well defined interface
- You should not know how it is implemented
  - implementation can e changed later
- The black box should not know how it will be used later
  - it can be used in many unexpected ways

Contract

- Interface and description can be seen as a contract
- Read the description
  - fillRect(x,y,h,w)
  - Not fillRect(x1,y1,x2,y2)

Static subroutines and variables

- Subroutine definition
- modifiers – static and public, private, protected
- return-type – void or typename
- parameter-list – next slide

```
⟨modifiers⟩ ⟨return-type⟩ ⟨subroutine-name⟩ ( ⟨parameter-list⟩ ) {
    ⟨statements⟩
}
```

```
public static void main(String[] args) { ... }
```

Calling subroutines

- Inside the class
  - playGame()
- Outside the class
  - Poker.playGame()
  - Integer.parseInt(“33”)

```
⟨subroutine-name⟩(⟨parameters⟩);
```

```
⟨class-name⟩.⟨subroutine-name⟩(⟨parameters⟩);
```

Subroutines in programs

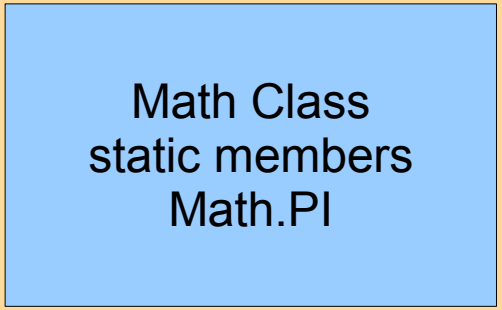
- Split problem into smaller parts
- Use the same subroutine in several places
- Simple main loop

```
public class GuessingGame {

   public static void main(String[] args) {
      TextIO.putln("Let's play a game.  I'll pick a number between");
      TextIO.putln("1 and 100, and you try to guess it.");
      boolean playAgain;
      do {
         playGame();  // call subroutine to play one game
         TextIO.put("Would you like to play again? ");
         playAgain = TextIO.getlnBoolean();
      } while (playAgain);
      TextIO.putln("Thanks for playing.  Goodbye.");
   } // end of main()
```

Member variables

- We only look at static member variables
  - for now
- These belong to the class not the individual object
- Example PI, which is also final



Static member variables

- Static does not mean final
- Local variables in subroutines
- Global variables in classes
  - can be public or private

```
static String usersName;
public static int numberOfPlayers;
private static double velocity, time;
```

Parameters

- You have called lost of methods with parameters
- These are called actual parameters
- Formal parameters
  - The one you write when you define a subroutine, that others can call
- Actual parameters are substituted for the formal ones

Parameters

- Formal parameters
  - only declared once

```
static void print3NSequence(int startingValue) {
```

- Actual parameters
  - as many times as you call the method
  - print3NSequence(17);

```
do {
   TextIO.putln("Enter a starting value;")
   TextIO.put("To end the program, enter 0: ");
   K = TextIO.getInt();  // Get starting value from user.
   if (K > 0)   // Print sequence, but only if K is > 0.
      print3NSequence(K);
} while (K > 0);   // Continue only if K > 0.
```

Subroutine example

```
static void doTask(int N, double x, boolean test) {
    // statements to perform the task go here
}
```

```
doTask(17, Math.sqrt(z+1), z >= 10);
```

```
{
  int N;       // Allocate memory locations for the formal parameters.
  double x;
  boolean test;
  N = 17;                 // Assign 17 to the first formal parameter, N.
  x = Math.sqrt(z+1);     // Compute Math.sqrt(z+1), and assign it to
                          //    the second formal parameter, x.
  test = (z >= 10);       // Evaluate "z >= 10" and assign the resulting
                          //     true/false value to the third formal
                          //     parameter, test.
   // statements to perform the task go here
}
```

Overloading

- Many methods with the same name
- TextIO example
  - putln(int)
  - putln(String)
  - putln(boolean)
- No overloading on return-type

Bad parameter values

- This is an error
- What do you do?
- Throw an exception

```
static void print3NSequence(int startingValue) {

   if (startingValue <= 0)  // The contract is violated!
      throw new IllegalArgumentException( "Starting value must be positive." );

   .
   .  // (The rest of the subroutine is the same as before.)
   .
```

If then else

- Leftover from lecture2
- If...then...else on a single line

```
⟨boolean-expression⟩ ? ⟨expression1⟩ : ⟨expression2⟩
```

```
next = (N % 2 == 0) ? (N/2) : (3*N+1);
```

Exercises

- Avoid warning
- write
  - private static final long serialVersionUID = 1L
- or click error message and select
  - Add default serial versionID