Reminder of Java

Chonnam National University
School of Electronics and Computer Engineering

Kyungbaek Kim

Java is

- Object Oriented
- Platform independent
- Simple
- Secure
- Architectural-natural
- Portable
- Robust
- Multi-threaded
- Interpreted

Java Basic Syntax

- Object
  - States and behaviors
  - e.g. a dog → state : color, name, breed
    → behavior : barking, eating
  - Instance of a class
- Class
  - A template (blue print) describing an object
- Methods
  - Basically a behavior
  - A class can contain many methods
- Instance Variables
  - Each object has its unique set of instance variables

Rules of Java - Hello world

```
public class MyFirstJavaProgram{
        /* This is my first java program named "MyFirstJavaProgram.java".
        * This will print 'Hello World' as the output
        */
        public static void main(String []args){
                System.out.println("Hello World"); // prints Hello World
        }
}
```

- Case Sensitive
  - **Hello** and **hello** is different in Java
- Class Names
  - First letter should be in ***Upper Case***
  - Other inner words should start with ***Upper Case***
- Method name
  - All method names should start with a ***Lower Case***
  - Other inner words should start with ***Upper Case***
- Program File Name
  - Should exactly match the class name
- public static void main(String []args)
  - Java program starts from here

Java Identifiers

- Names for classes, variables and methods
  - Should begin with a letter (A to Z or a to z), currency character ($) or underscore (_)
  - After the first character, any combination of characters are possible
  - A keyword can not be used as an identifier
  - Case sensitive
  - Legal identifiers : age, $salary, _value, _1_value
  - Illegal identifiers : 123abc, -salary

Package

- Package
  - Groups of java classes and java interfaces
  - A tool for managing a large name space and avoiding conflicts
- Package names
  - Consists of words separated by periods
  - Package name also reflects its directory structure
    - e.g. dnslab.ood.homework1
      → /javahome/dnslab/ood/homework1
- Importing a class from a package
  - e.g. import dnslab.ood.homework1.TetrisGame
  - e.g. import java.lang.String or java.net.*

CLASSPATH

- Environment variable must be set to point to any directories that contain java classes and packages
  - e.g. setenv CLASSPATH ".:/user/share/java/classes"
  - e.g. java –classpath . TetrisGame
- JAR file
  - An archiving file of java classes
  - Used to set classpath
    - e.g. java –classpath ./Tetris.jar TetrisGame
  - "-jar" can be used
    - e.g. java –jar Tetris.jar
  - How to create jar file ?
    - Use jar command
      - e.g. jar cvf classes.jar –C foo/ .
    - Easier way? → GUI based jar exporting

Java Variables

- Local Variables
- Class Variables
  - static
- Instance Variables
  - Non static variables

```
package new_tetris;
import new_tetris.intStuff.*;

class TetrisGame implements PosPaintable {
        Grid playfield;
        TetrisPiece piece;

        public TetrisGame(int xs,int ys,boolean randomCrap){

                String msg;
                playfield=new Grid(xs,ys,1);
                clear();
                piece = new TetrisPiece();
        }
....
```

Java Arrays

```
// Allocate array in the heap, a points to the array
// reference
int[] a = new int[100];
a[0] = 13;
a[1] = 26;
// read-only access .length → 100
a.length
// throws “ArrayOutOfBounds” exception at runtime
a[100] = 13
```

Multidimensional Arrays

- An array with two or more dimensions
  - e.g. int[][] grid = new int[100][100]
- Specify two indexes to refer to each element
  - e.g grid[0][1] = 10;

```
int temp;
// 10x20 2-d array
int[][] grid = new int[10][20];
grid[0][0] = 1;
grid[9][19] = 2;
temp = grid.length; // 10
temp = grid[0].length; // 20
grid[0][9] = 13;
// really it's just a 1-d array
int[] array = grid[0];
temp = array.length; // 20
temp = array[9]; // 13
```

Java Enums

- Restrict a variable to have on of only a ***few predefined values***
  - Introduced in Java 5.0
  - Possible to reduce the number of bugs

```
class FreshJuice{
        enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
        FreshJuiceSize size;
}

public class FreshJuiceTest{
        public static void main(String args[]){
                FreshJuice juice = new FreshJuice();
                juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
        }
}
```

Data types in Java

- Primitive Data Types
  - byte, short, int, long, float, double, boolean, and char
- Reference/Object Data Types
  - Created using defined constructors of the class
    - Class objects and various type of array variables
  - Default value of any reference variable is null
    - e.g. Animal animal; animal = new Animal(“lion”);

Access Modifiers

- Default
  - Visible to the package
- Private
  - Visible to the class only
- Public
  - Visible to the world
- Protected
  - Visible to the package and all subclasses

Java Methods

- A collection of statements that are grouped together to perform an operation

```
modifier returnValueType methodName(list of parameters) {
// Method body;
}
```

Exception Handling

- Try and Catch
  - Method can throws “exceptions”

```
try
{
        //Protected code
}catch(ExceptionType1 e1)
{
        //Catch block
}catch(ExceptionType2 e2)
{
        //Catch block
}catch(ExceptionType3 e3)
{
        //Catch block
}
```

Basic Operators

- Arithmetic Operators
  - +, -, *, /, %, ++, --
- Relational Operators
  - ==, !=, >, <, >=, <=
- Bitwise Operators
  - &, |, ^, ~, <<, >>, >>>
- Logical Operators
  - &&, ||, !
- Assignment Operators
  - =, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=
- Conditional Operators
  - Variable x = (expression)? Value if true: value if false
- InstanceOf Operator
  - Check whether the object is of a particular type (Class of Interface)

While loop and do while loop

- While
  - e.g. while(boolean_expression){
    //statements
    }
- Do While
  - e.g. do{
    //statements
    }while(boolean_expression)

For loops

- For
  - e.g. for(initialization; boolean_expression; update){
    //statement
    }
- Iterator can be used
  - Userd for Containers : vector, list, map, set
  - e.g. while(iterator.hasNext()){
    //statement
    }

Break and Continue

- Break
  - To stop the entire loop
  - Must be used inside any loop or a switch statement
- Continue
  - To immediately jump to the next iteration of the loop

If and if…else

- if
  - e.g. if(boolean_expression){
    //statement if the expression is true
    }
- If…else
  - e.g. if(boolean_expression){
    //statement if the expression is true
    }else{
    //statement if the expression if false
    }
- If…else if…else

switch

```
switch(expression){
        case value :
                //Statements
                break; //optional
        case value :
                //Statements
                break; //optional
                //You can have any number of case statements.
        default : //Optional
                //Statements
}
```

Inheritance

Dad's smile

Mom's eyes

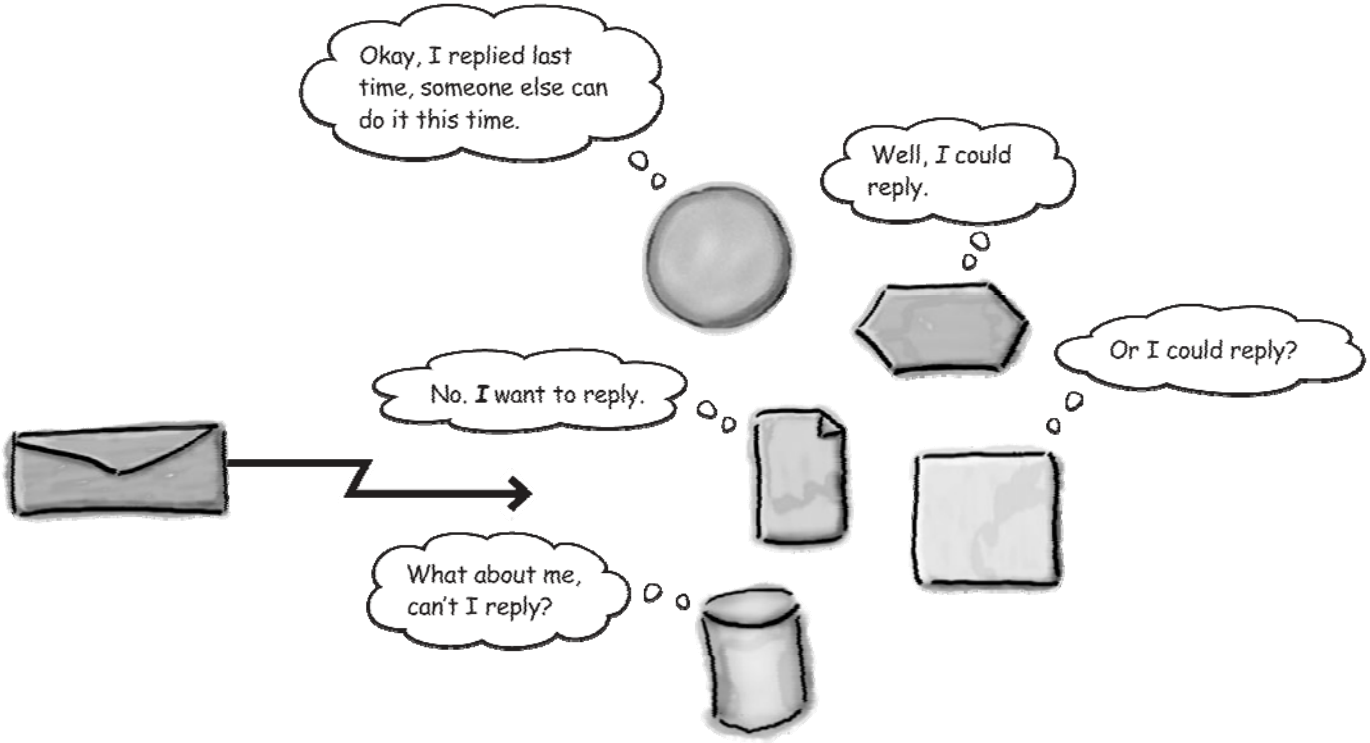
Dad's sports obsession

Mom's love of ROCK

```
public class Son extends Mom, Dad {
        doSoccor()
        playGuitar()
}
```

Polymorphism

- Different type of objects can respond to the same message
- The actual method that executes is not determined until run time
  - Dynamic (or late) binding



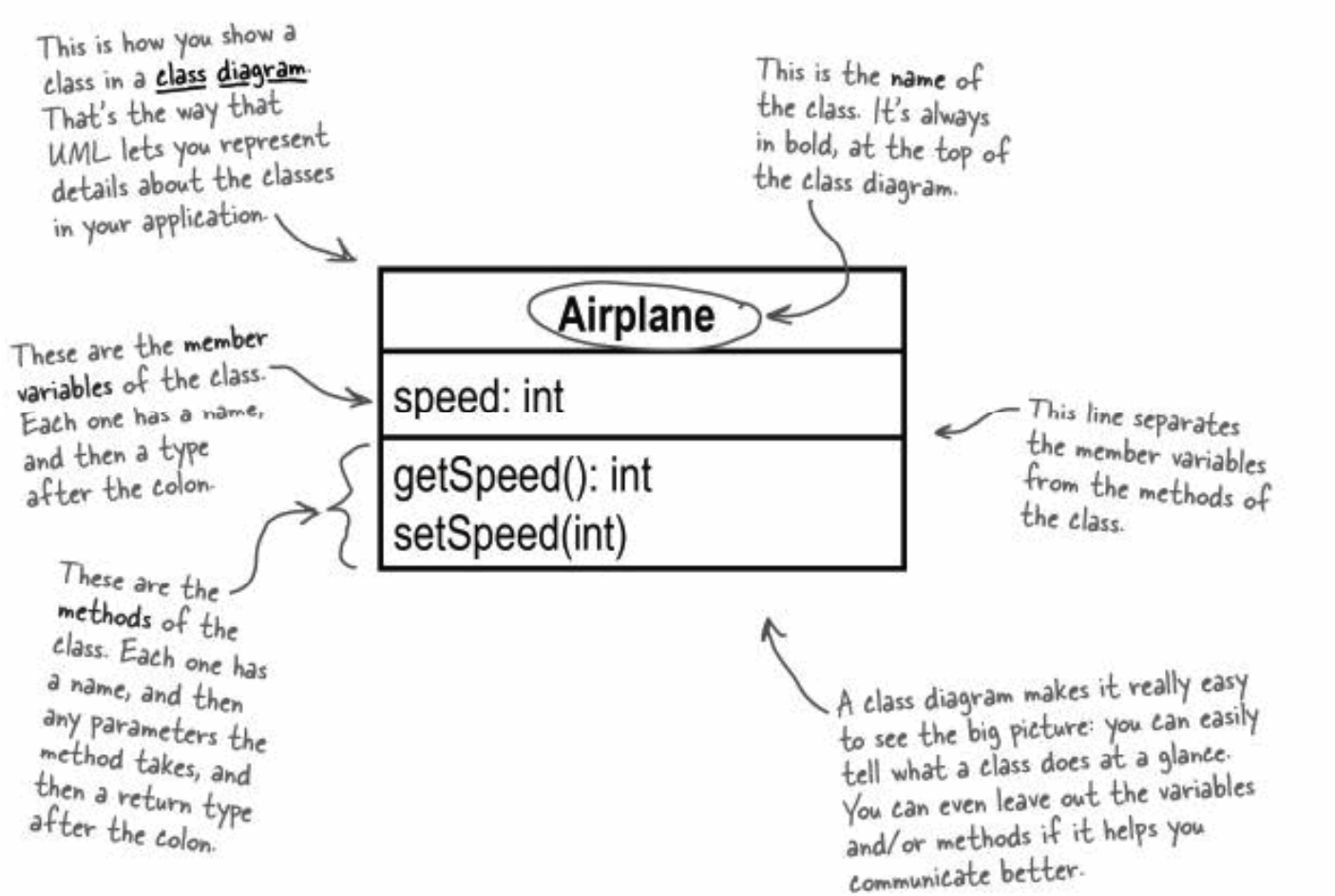
Encapsulation

Encapsulation is when you protect information in your code from being used incorrectly.

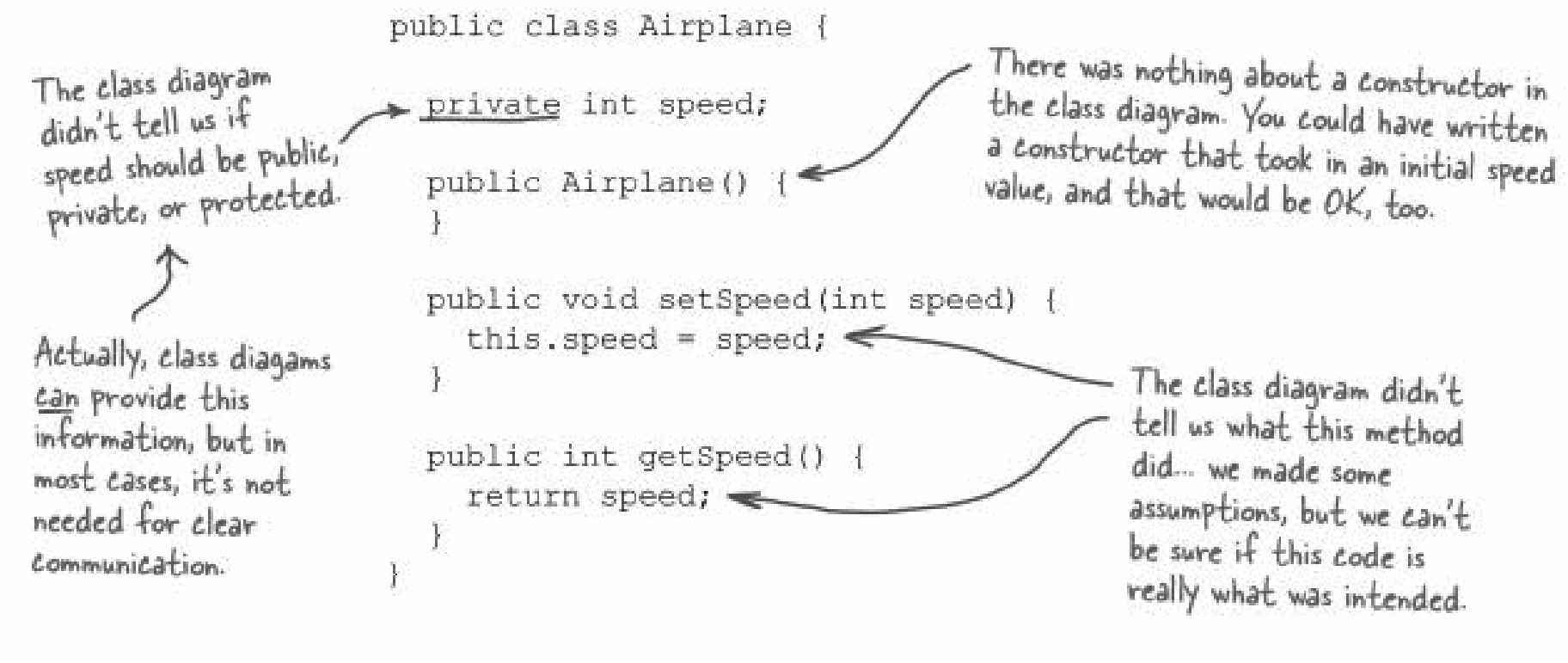
One of the biggest advantages of O-O is the ability to make changes to an object's implementation without affecting other parts of the program.

Intro of UML

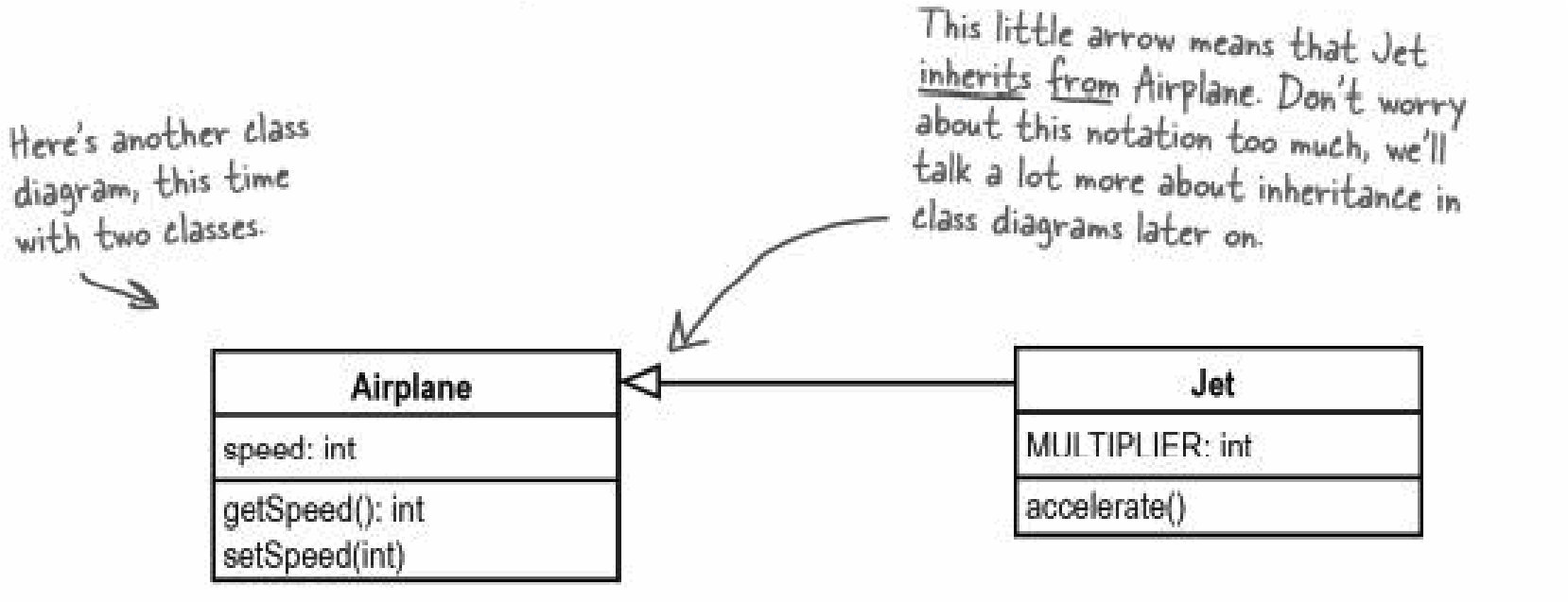
- Unified Modeling Language



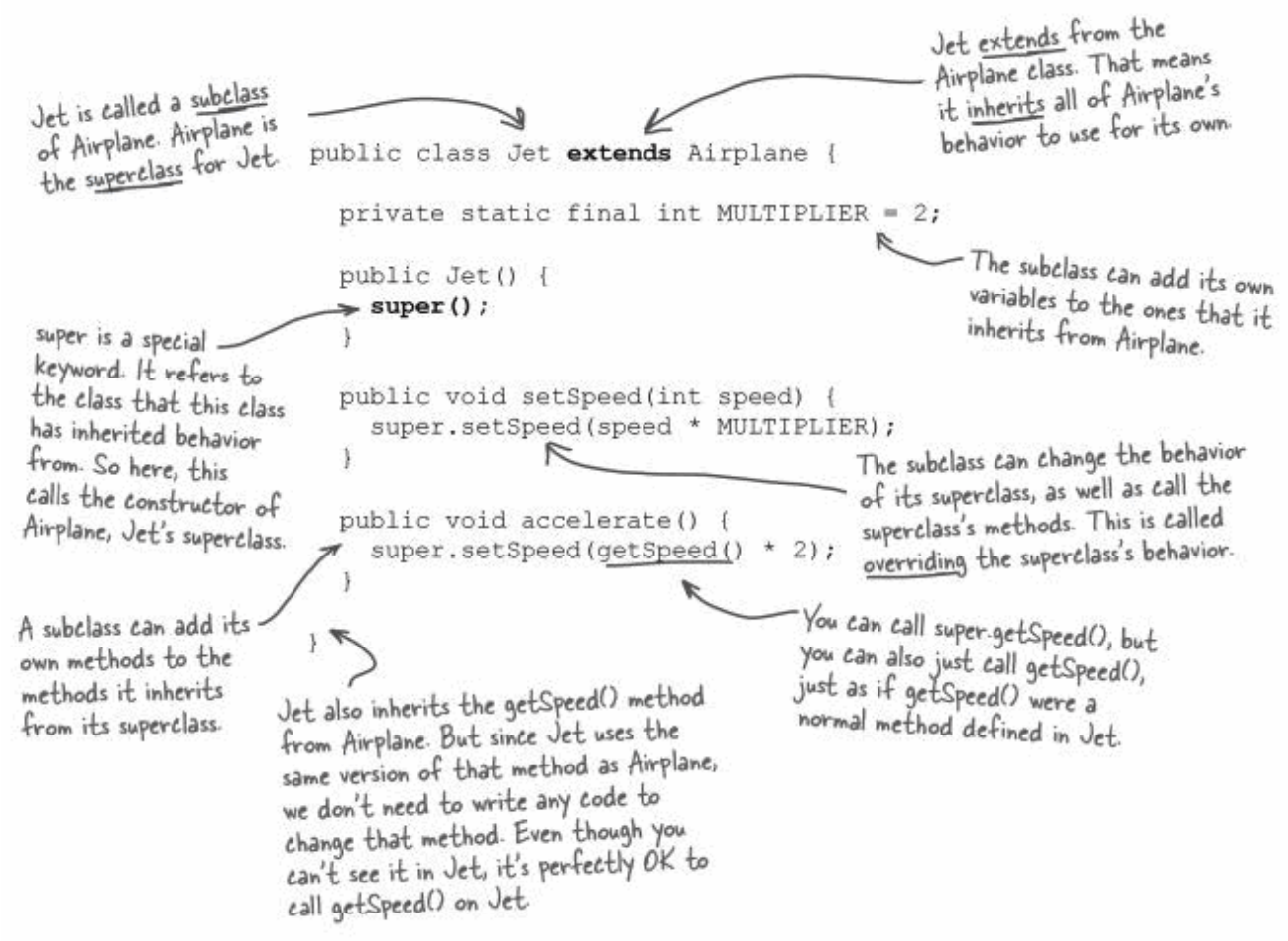
Example of Java Class



UML representation of Inheritance and Polymorphism



Example of Inheritance



Example of Polymorphism