COE 318
Introduction to Software Design

Lecture 1

Hamed Karimi

Trustworthy AI
Research Lab
tailab.org

Toronto
Metropolitan
University

What do we learn in this course?

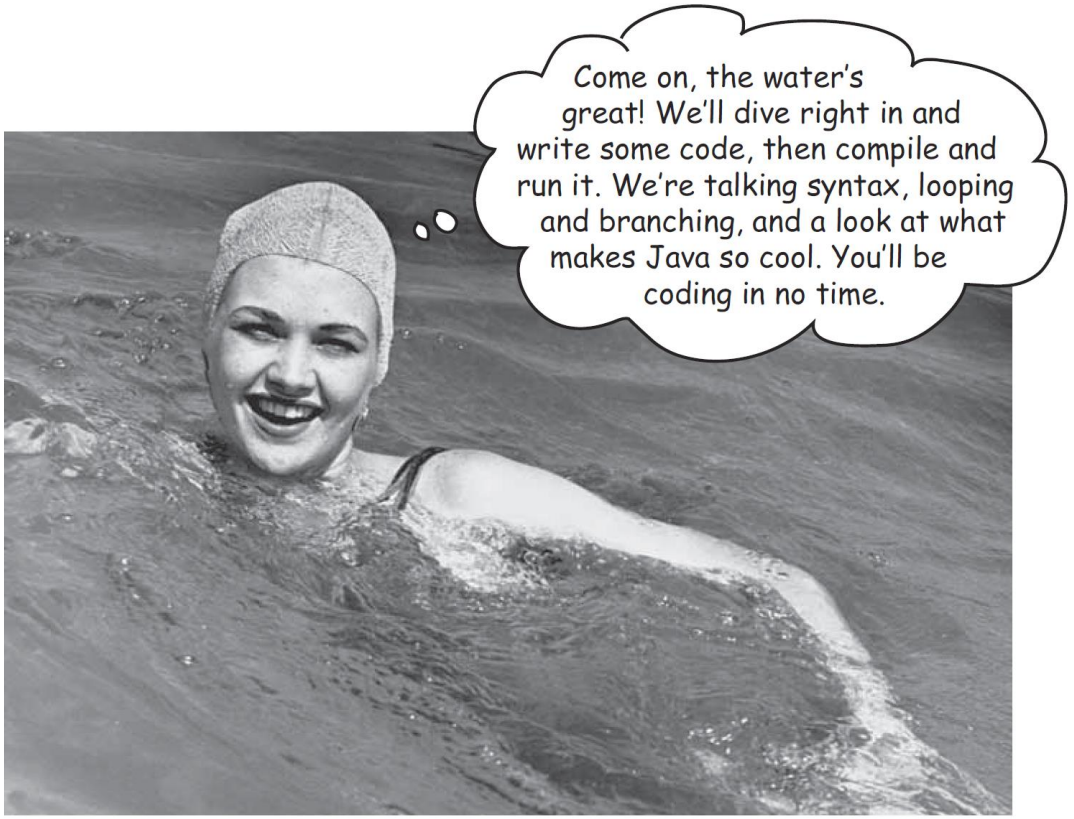
- Software Development Lifecycle (brief)
- Object-oriented programming (Java)
  - Object, Class
  - Encapsulation, Inheritance, Polymorphism
- Practice, practice, practice – this is a very hands-on course!

Evaluation
(find final outline in D2L)

Theory	
Midterm Exam	30 %
Final Exam	40 %
Laboratory	
Lab Reports + Lab Quiz	30 %
TOTAL:	100 %

Textbook

- "Head First Java", Kathy Sierra & Bert Bates, second edition



Programming Language

- A programming language is a high-level language that contains instructions that controls a computer's operations.
- Examples: Java, C++, C, Python
- Compiling: A programming language needs to be translated into a low-level machine code before execution on a computer.

Software Development Lifecycle (SDLC)

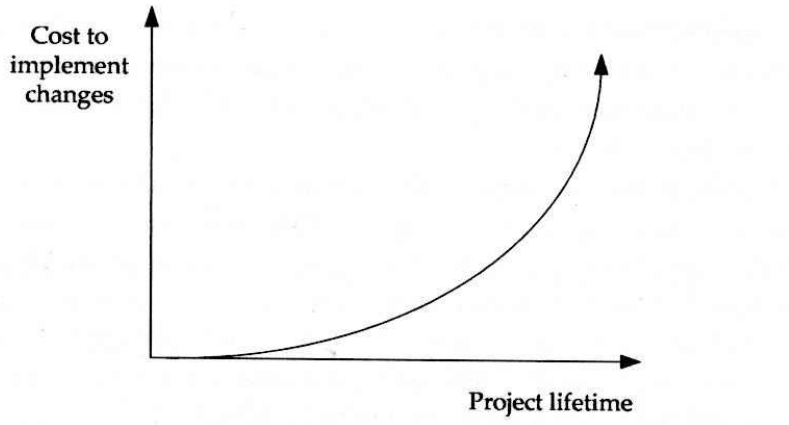
- Series of activities to develop software:
  - Requirement gathering and analysis
  - Design
  - **Implementation/Coding**
  - **Testing**
  - Deployment
  - Maintenance

SDLC Models

- Waterfall model
- Iterative model
- Agile model
- V Model
- RAD model
- Spiral model

A major software development challenge

- Cost of design changes (increased) exponentially as the project lifetime increases



Slide adopted from Design for Electrical and Computer Engineers, Ford & Coulston (2008)

Interactions between software systems and sociocultural contexts

- Human lives are increasingly dependent on technology and particularly Software Systems
- Software Systems are also increasingly playing roles in human decision making
- Thus, daily **Social** and **Individual** experiences of human beings are influenced by Software Systems
- Examples…

Software problems

- 2016: Software glitch causes F-35 to incorrectly detect targets information
  - F-35s detect a single ground threat such as anti-aircraft weaponry. The sensors on the planes may have trouble distinguishing whether it was an isolated threat or several objects

https://www.tricentis.com/software-fail-watch/
https://www.foxnews.com/tech/software-glitch-causes-f-35-to-incorrectly-detect-targets-in-formation

Software problems

- 2020: Boeing Fixing New Software Bug on Max; an indicator light, designed to warn of a malfunction by a system that helps raise and lower the plane's nose.
- It was turning on when it wasn't supposed to

How the Boeing 737 Max Disaster Looks to a Software Developer
https://spectrum.ieee.org/aerospace/aviation/how-the-boeing-737-max-disaster-looks-to-a-software-developer

https://www.bloomberg.com/news/articles/2020-02-06/boeing-identifies-new-software-problem-on-grounded-737-max-jet

Software problems

- 2021- Racial bias in AI systems: Most AI datasets for human/face recognition are not racially diverse, resulting in bias when deployed.
- Facebook apologizes after AI puts ‘Primates’ label on videos of Black Men

https://www.nytimes.com/2021/09/03/technology/facebook-ai-race-primates.html

Software problems

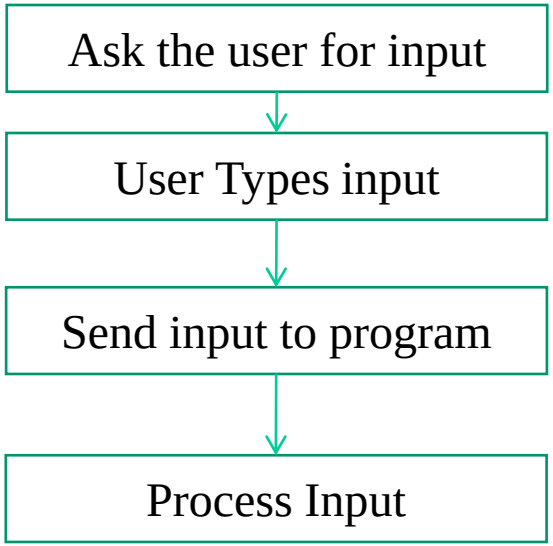
- Bias in hiring software (commonly used by many companies nowadays)
- https://www.businessinsider.com/amazon-built-ai-to-hire-people-discriminated-against-women-2018-10

Ethics and Professionalism in Software Development

- As Software Systems are becoming the fabric of society, software developers need to be conscious of the ethical implications of what they develop
- There are big ethical issues in today's software development, for example
  - Promoting or demoting **digital divide**
  - **Protecting or violating individual privacy**
  - **Supporting or infringing human rights, copyrights**
  - **…**
- These issues are not only about big software developers such as Google, Facebook, Amazon but the ethics needs to be embedded in all the small decisions we constantly make in our software development efforts in every level
- We should not use technology as an excuse
- We should own the programs we are developing and choose ethically

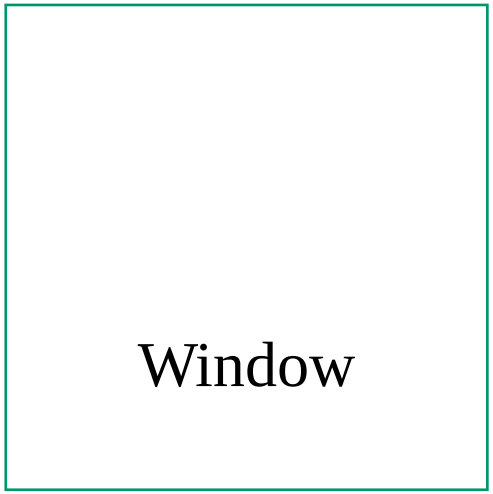
Language Paradigms

- Structured – Top to bottom. Programming is done sequentially.



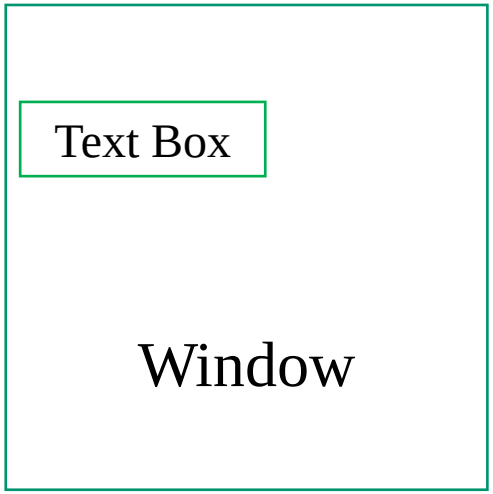
Language Paradigms

- Object-oriented – define different “objects” first, what they do etc. Then link them together.



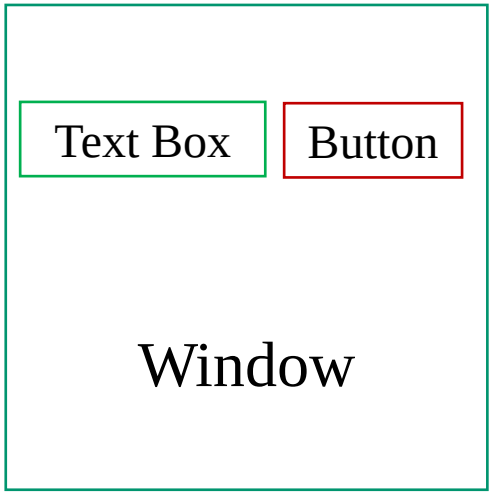
Language Paradigms

- Object-oriented – define different “objects” first, what they do etc. Then link them together.



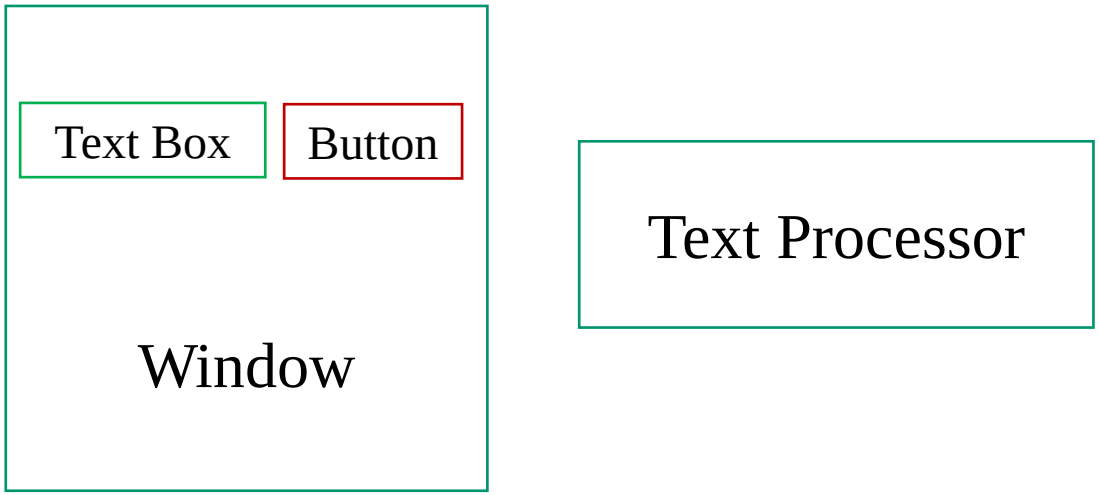
Language Paradigms

- Object-oriented – define different “objects” first, what they do etc. Then link them together.



Language Paradigms

- Object-oriented – define different “objects” first, what they do etc. Then link them together.



Language Paradigms

- Structured – easier to understand. Difficult to manage large projects, where many developers contribute to different components.
- Object-oriented – easier to reuse objects for larger projects.
- A lot of modern languages are multi-paradigm (e.g. JavaScript).

Objects and Classes

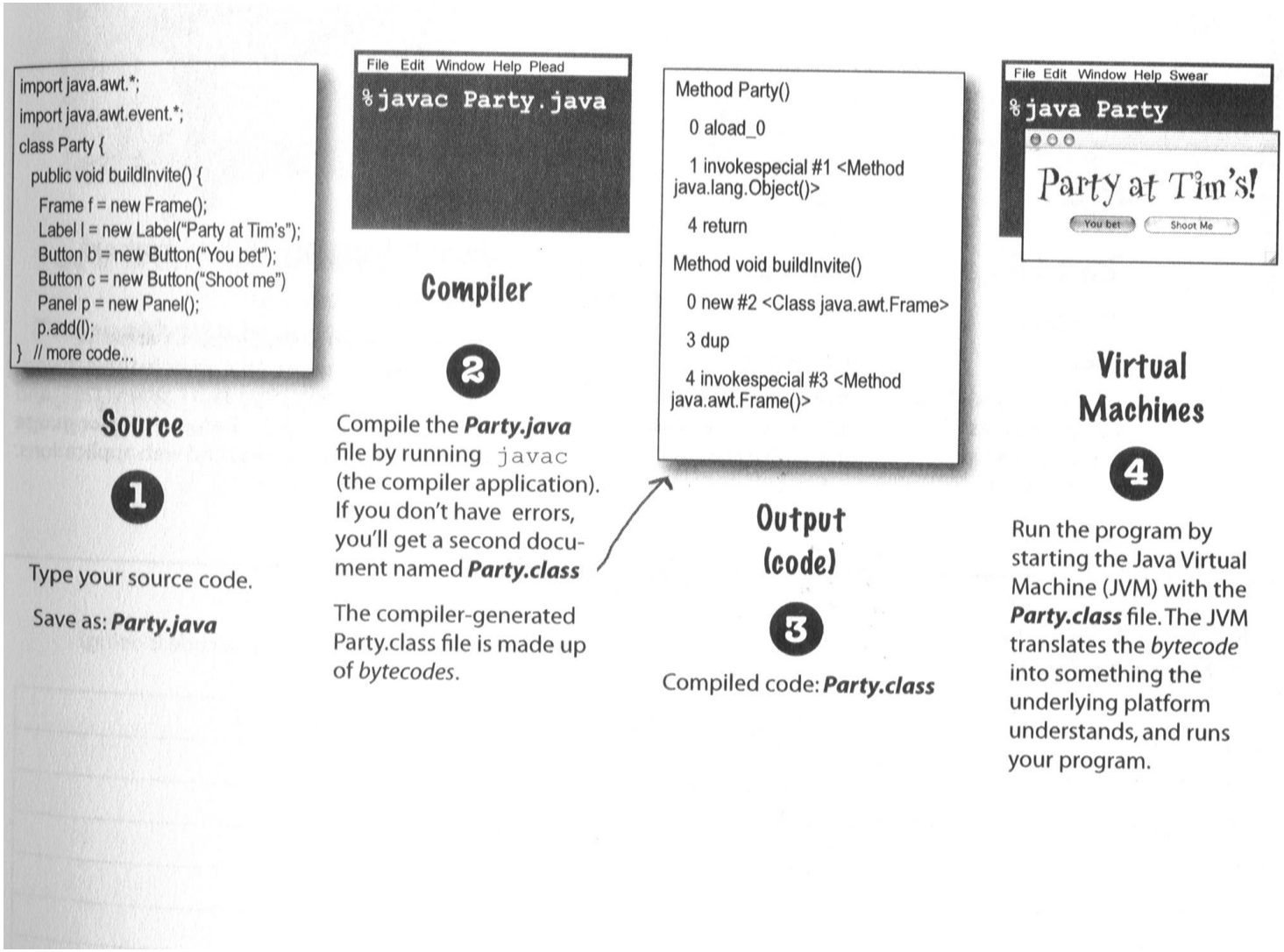
- Objects: “things” that have common states and behaviors.
- Class: A “template” that defines an object.

Real world Objects	*State*	*Behavior*
Dog	Name Color Hungry	Barking Fetching Wagging tail
Bicycle	Current gear Current pedal cadence Two wheels Speed	Braking Accelerating

Java

- Simple
- Object-oriented
- Platform Independent
- Safe. No pointers. Live in virtual machine.
- Multi-threaded
- Garbage collected

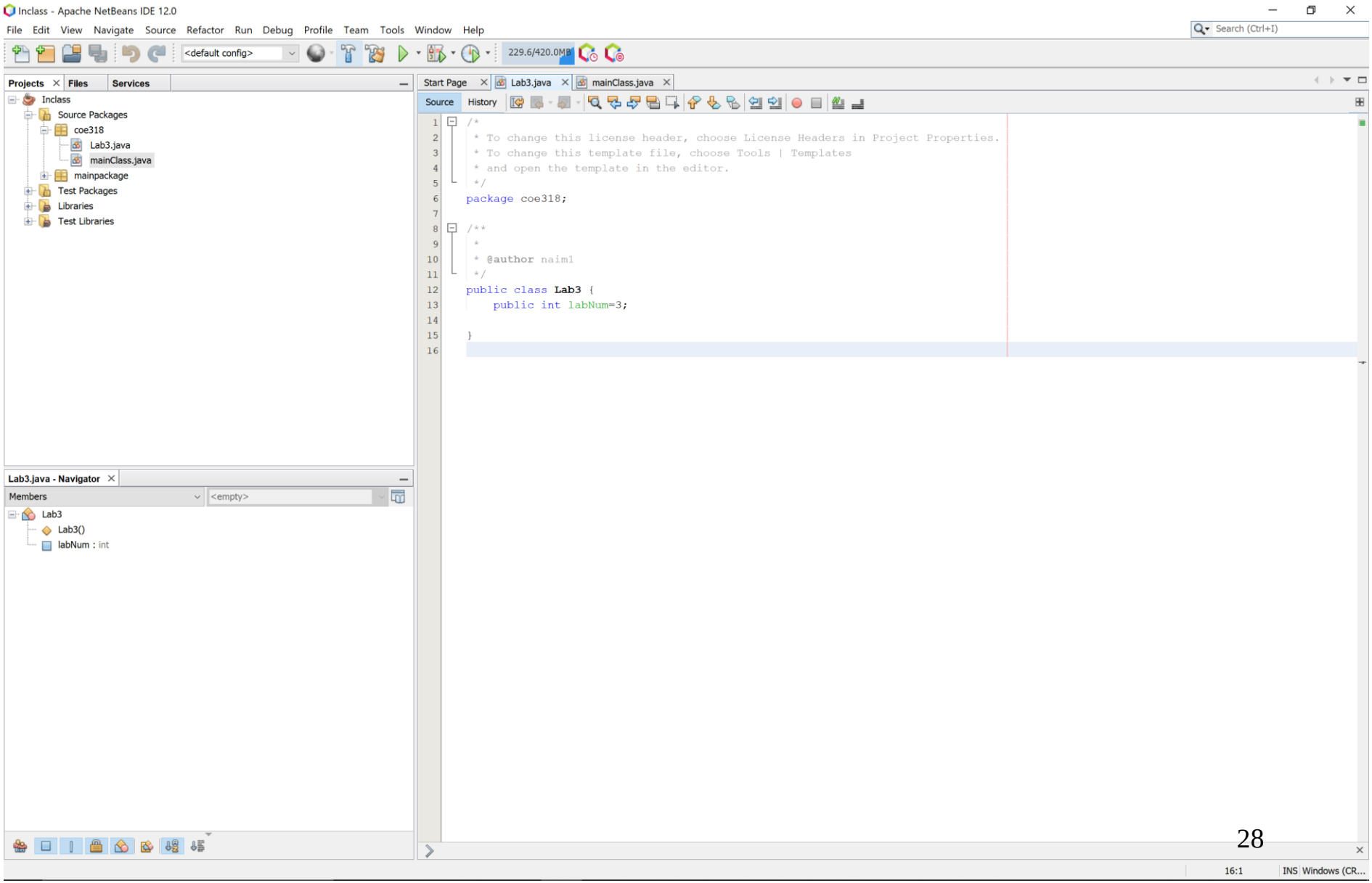
How Java Works



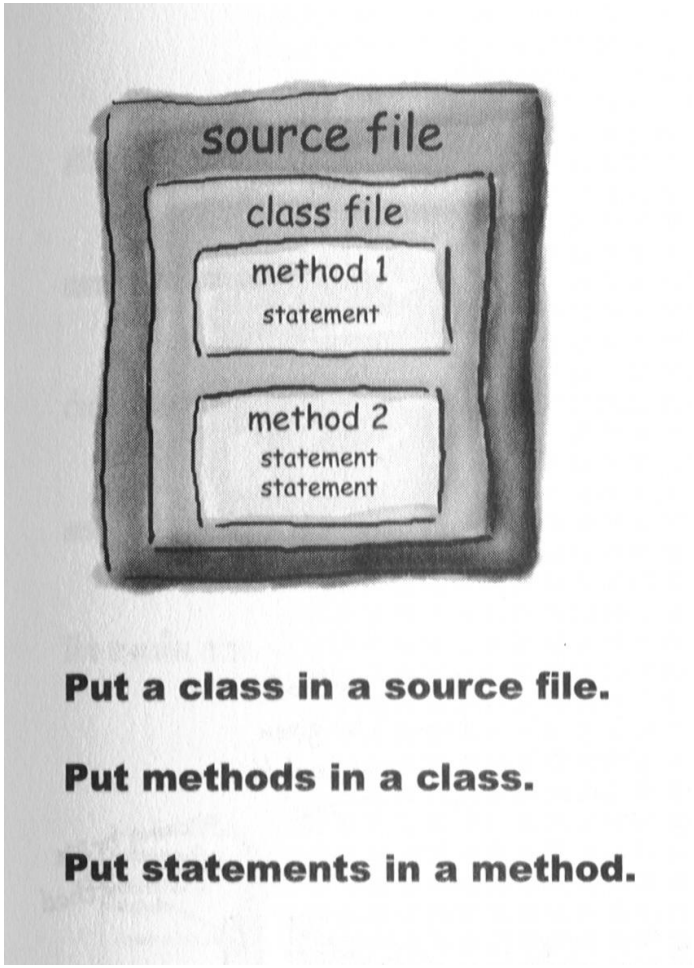
Netbeans – Java IDE

- NetBeans: Java Integrated Development Environment (IDE)
- Free download from https://netbeans.apache.org/

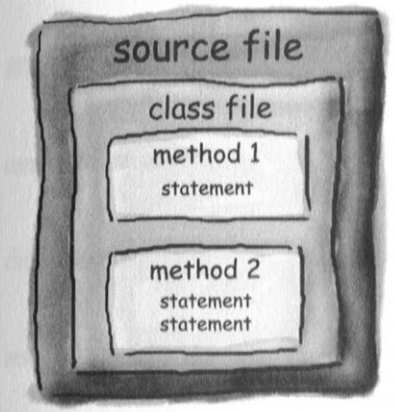
Netbeans – Java IDE



Java Code Structure



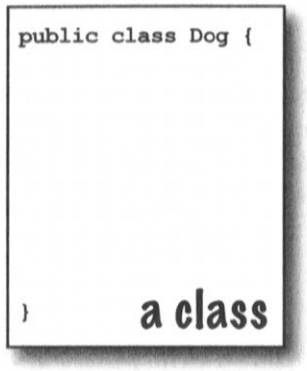
Java Code Structure



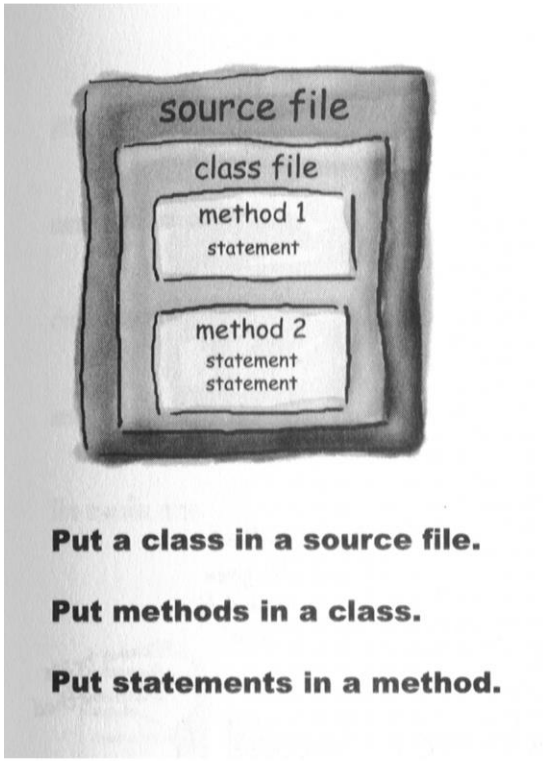
Put a class in a source file.

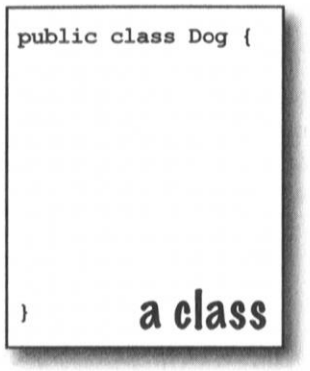
Put methods in a class.

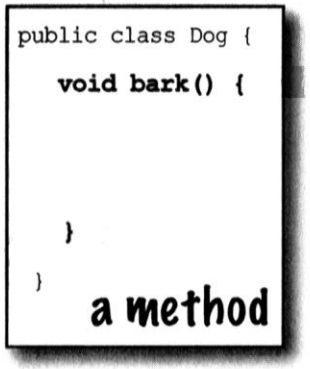
Put statements in a method.

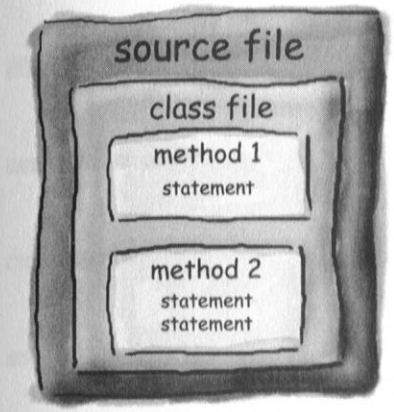


Java Code Structure









Put a class in a source file.

Put methods in a class.

Put statements in a method.

```
public class Dog {

}
```

a class

```
public class Dog {
   void bark() {

   }
}
```

a method

```
public class Dog {
   void bark() {
     statement1;
     statement2;
   }
}
```

statements

Hello World in Java

```
public class HelloWorld
{
    public static void main(String[] args)
    {
            System.out.println("Hello World!");
    }
}
```

Declaration

- A variable is like a container that can hold certain content/value.
- Variable declaration consists of type and name
- The type shows what content the “container” can hold.
- The variable name will be used in other statements when such content is desired.

```
int age;
float weight;
String name;
```

Assignment

- Assignment is to put content into a container.
- Assignment syntax is variable = value;

```
age=20;
weight=145;
name="Dustin";
```

Comments

- End of line (single line) comment starts with //
- Multiple line comments are enclosed by /* and */

```
//single line comment
age=20;
/* multiline
Comment*/
weight=145;
name="Dustin";
```

Demo StatementExample.java

```
public class StatementExample
{
     public static void main(String[] args)
    {
            int age;
            float weight;
            age = 5; //My dog is five years old.
            System.out.println("My dog is " + age + "years old.");
            weight = 10; //My dog weighs 10 pounds.
           System.out.println("My dog weighs " + weight + " pounds.");
           }
}
```

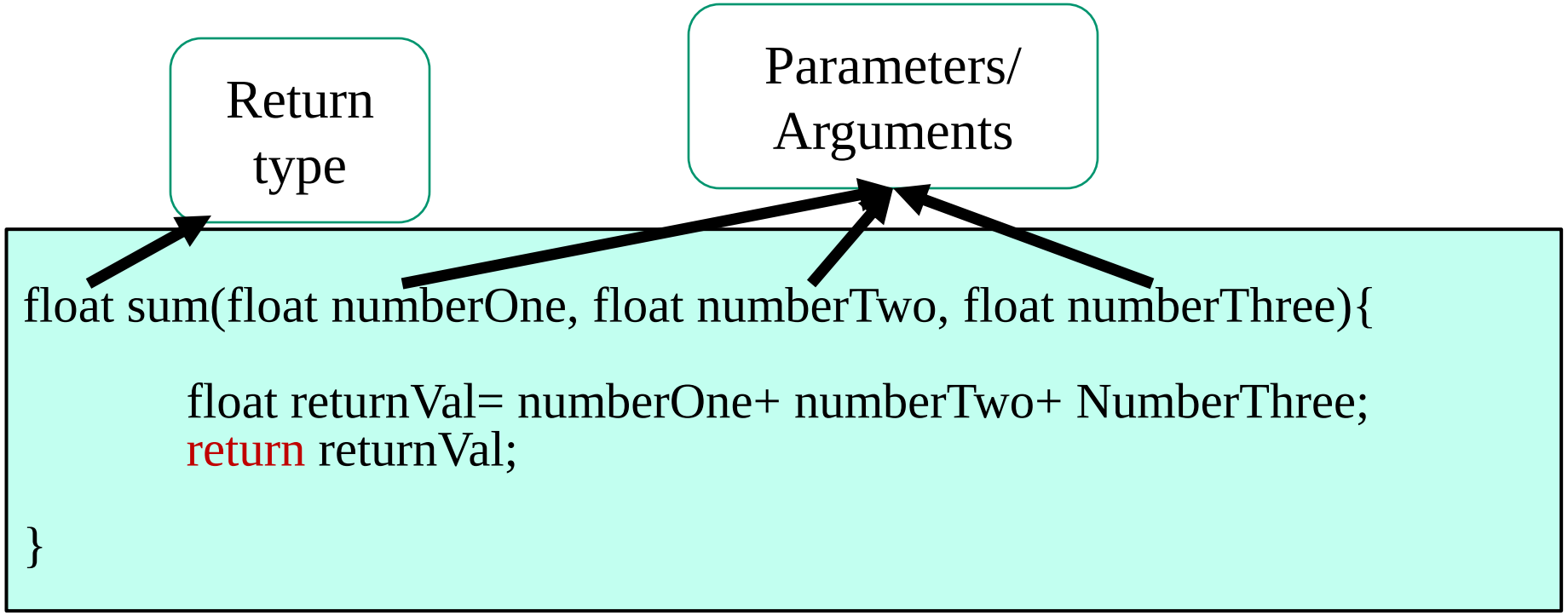
Console Output

```
System.out.print("Welcome to COE318!");
System.out.print("Hope it will be fun!");
```

```
System.out.println("Welcome to COE318!");
System.out.println("Hope it will be fun!");
```

Method

- A method is a unit of work always done together.
- Call the method when you want to do such a unit of work.
- Methods have return types and parameters



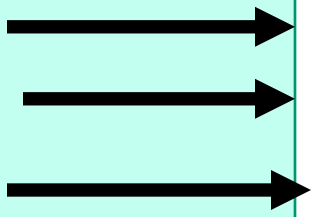

```
//To call the above function in the code

float result = sum( 23.0, 24.0, 25.0);
```

Class

- A “template” that defines type of an object.
- Can contain - variables, methods.

```
public class Car {
     private String licensePlate;
     private double speed;
     private double maxSpeed;

    public Car(String plateNumber, double speedVal, double maxVal){
    }
    public void setLicensePlate(String plateNumber ){
     }
    public void setSpeed (double speedVal) {
     }
     public double getSpeed () {
     }
}
```

```
public class Car {
     private String licensePlate;
     private double speed;
     private double maxSpeed;

    public Car(String plateNumber, double speedVal, double maxVal){
    }
    public void setLicensePlate(String plateNumber ){
     }
    public void setSpeed (double speedVal) {
     }
     public double getSpeed () {
     }
}
```

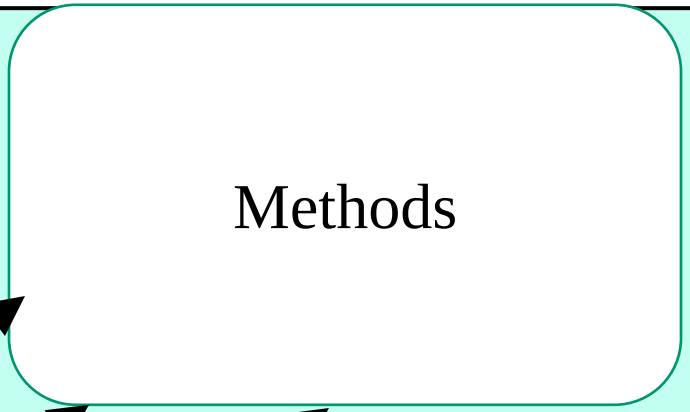


Variables
(also called instance variables)

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

   public Car(String plateNumber, double speedVal, double maxVal){
   }
   public void setLicensePlate(String plateNumber ){
    }
   public void setSpeed (double speedVal) {
    }
    public double getSpeed () {
    }
}
```

Access modifiers (will be explained later)

```
public class Car {
     private String licensePlate;
     private double speed;
     private double maxSpeed;

    public Car(String plateNumber, double speedVal, double maxVal){
    }
    public void setLicensePlate(String plateNumber ){
     }
    public void setSpeed (double speedVal) {
     }
     public double getSpeed () {
     }
}
```



```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

   public Car(String plateNumber, double speedVal, double maxVal){
   }
   public void setLicensePlate(String plateNumber ){
    }
   public void setSpeed (double speedVal) {
    }
    public double getSpeed () {
    }
}
```

Constructor. A special "method" that is used when creating objects from this class (Notice same name "Car" as the class)

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

 public Car(String plateNumber, double speedVal, double maxVal){
            licensePlate=plateNumber;
            speed=speedVal;
            maxSpeed=maxVal;
   }
 public void setLicensePlate(String plateNumber ){
            licensePlate=plateNumber;
    }
   public void setSpeed (double speedVal) {
            speed=speedVal;
    }
    public double getSpeed () {
            return speed;
    }
}
```

Object

- A class is simply a definition.
- To use a class, you need to create an object of that specific class type.
- We will use the Car class to create a Car object.
- Still need a main function to start things off from.

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

 public Car(String plateNumber, double speedVal, double maxVal){
                licensePlate=plateNumber;
                speed=speedVal;
                maxSpeed=maxVal;
   }
 public void setLicensePlate(String plateNumber ){
                licensePlate=plateNumber;
    }
   public void setSpeed (double speedVal) {
                speed=speedVal;
    }
    public double getSpeed () {
                return speed;
    }

 public static void main(String[] args) {

        Car c= new Car("C142",30,130);
        System.out.println(c.getSpeed());

   }

}
```

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

 public Car(String plateNumber, double speedVal, double maxVal){
                licensePlate=plateNumber;
                speed=speedVal;
                maxSpeed=maxVal;
   }
 public void setLicensePlate(String plateNumber ){
                licensePlate=plateNumber;
    }
   public void setSpeed (double speedVal) {
                speed=speedVal;
    }
    public double getSpeed () {
                return speed;
    }
}
```

The “new” keyword can create an object from a class definition. Notice the use of the “constructor” here

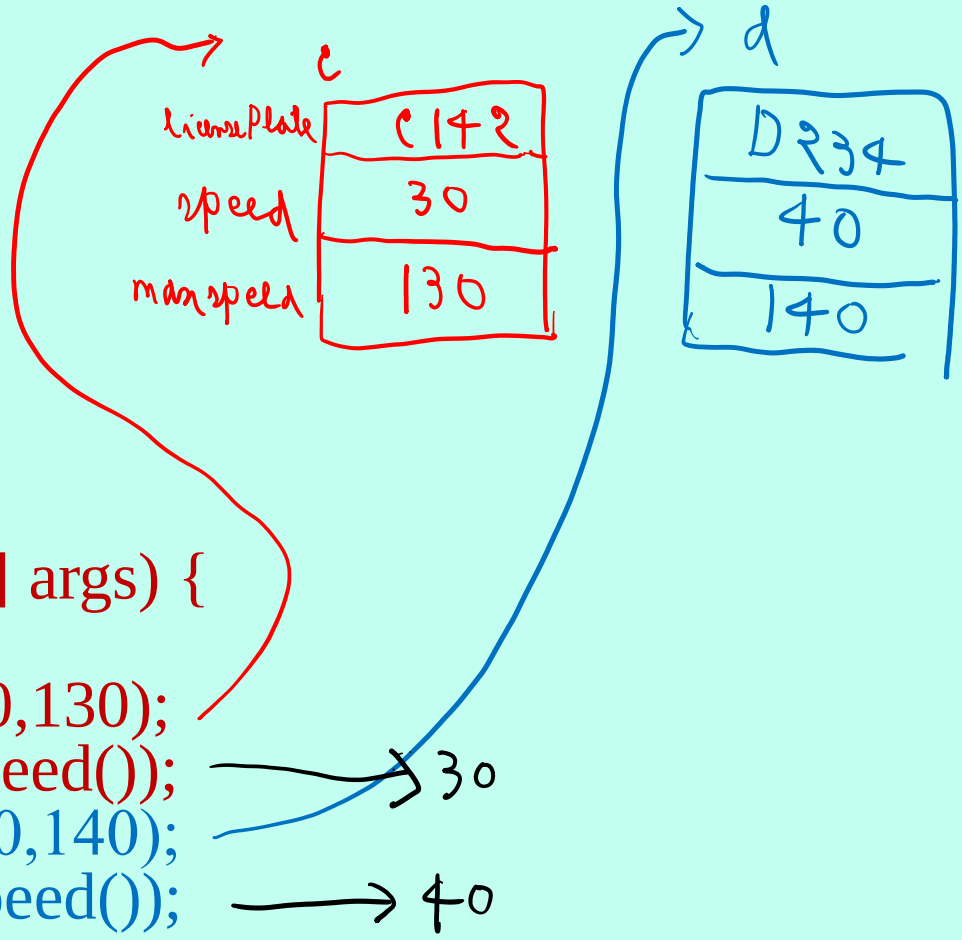
```
public static void main(String[] args) {

        Car c= new Car("C142",30,130);
        System.out.println(c.getSpeed());


   }
}
```

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

 public Car(String plateNumber, double speedVal, double maxVal){
                licensePlate=plateNumber;
                speed=speedVal;
                maxSpeed=maxVal;
   }
 public void setLicensePlate(String plateNumber ){
                licensePlate=plateNumber;
    }
   public void setSpeed (double speedVal) {
                speed=speedVal;
    }
    public double getSpeed () {
                return speed;
    }
}
```

The “.” operator can access specific variables/methods of an object

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

        Car c= new Car("C142",30,130);
        System.out.println(c.getSpeed());


   }
}
```

```
public class Car {
    private String licensePlate;
    private double speed;
    private double maxSpeed;

public Car(String plateNumber, double speedVal, double maxVal){
                licensePlate=plateNumber;
                speed=speedVal;
                maxSpeed=maxVal;
  }
public void setLicensePlate(String plateNumber ){
                licensePlate=plateNumber;
   }
  public void setSpeed (double speedVal) {
                speed=speedVal;
    }
    public double getSpeed () {
                return speed;
    }

public static void main(String[] args) {

        Car c= new Car("C142",30,130);
        System.out.println(c.getSpeed());
        Car d= new Car("D234",40,140);
        System.out.println(d.getSpeed());

    }

}
```



Good Programming Practices

- Start with a working Java program, modify it, break it, learn what you have broken it, then fix it.
- Use NetBeans to help you to learn (auto complete is amazing!).
- Use Blank lines and space characters for readability.
- Indent the content between { and }
- Make class/variable names meaningful.

Class/variable/method naming convention

- CamelCase Convention
- Class name's start with a capital letter, each subsequent word in identifier with Capital letter, e.g., HelloWorld, PersonalProfile.
- Variable and method names begin with a lowercase letter, and every word after first word begins with a capital letter, e.g., firstName, getName().

Make class/variable names meaningful

```
float gradeOne = 90.0;
float gradeTwo = 95.0;
float gradeThree = 85.0;
float averageGrade = (gradeOne + gradeTwo + gradeThree)/3;
```

```
float asdfghjk = 90.0;
float rujflwe = 95.0;
float asdfkj = 85.0;
float dfiefjek = (asdfghjk + rujflwe +asdfkj)/3;
```

Common Programming Errors

- Java is case sensitive. Pay attention to the Keyword, Identifier. E.g., most of the keywords are lowercases: “class”, “public”, “main”
- Java source file name should be the public class name plus “.java”: “HelloWorld.java”
- Semicolon at the end of a statement.
- Braces come in matching pairs. { and }