CSE 1801
Object Oriented Programming

Introduction

Acknowledgement

- For preparing the slides I took materials from the following sources
  - Course Slides of Dr. Tagrul Dayar, Bilkent University
  - Java book "*Java Software Solutions*" by Lewis & Loftus.

Outline

- Course Objectives
- Text Book
- Objects
- Classes
- Abstractions
- Encapsulations

Course Objectives

- Learning object-oriented programming with Java.
- Writing and enhancing
  - Classes
  - Arrays
  - Inheritance and polymorphism
  - Abstract classes and interfaces
  - Graphical user interface
  - I/O streams
  - Exceptions.

Text Book

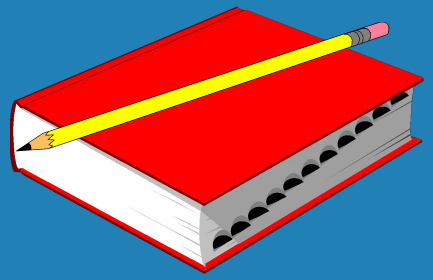
- **Lewis & Loftus, "Java Software Solutions – Foundations of program design", Addison Wesly, 8th edition, 2014.**
- **Deitel & Deitel, " Java: How to program"**

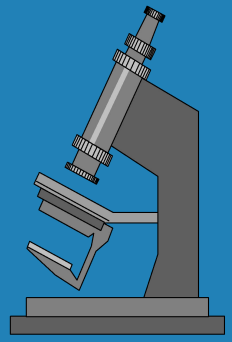
Java is an Object-Oriented Language

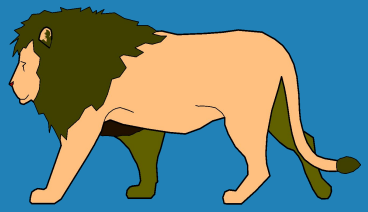
- In structured programming languages, methods define the structure of the programs, they are basic building blocks
- Data has secondary role, it is just something that is passed around.

- In object oriented languages, the data has the principal role
- Methods belong to the data, without the data, the method does not have any meaning (Except static methods)
- Data and methods together make up the object.
- OOP tries to model the real world.
- What does the real world look like?

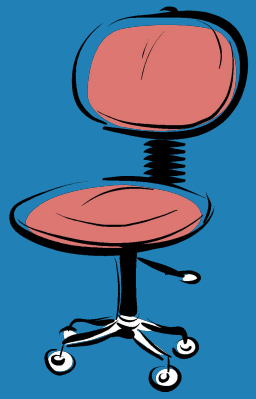
Objects everywhere...

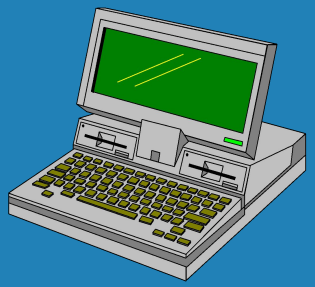




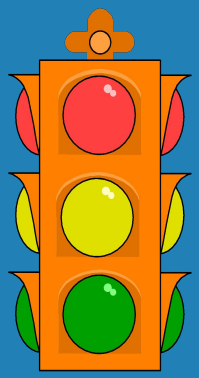


Real world entities

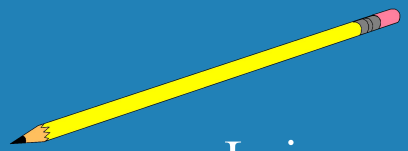




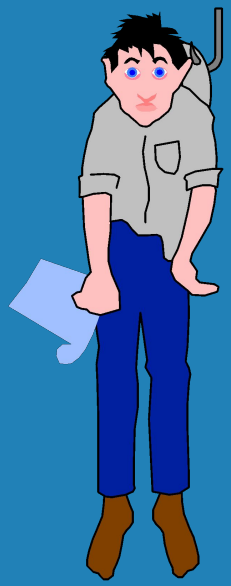
Objects have state...



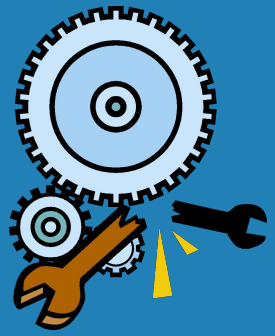
Red



Lyin
g

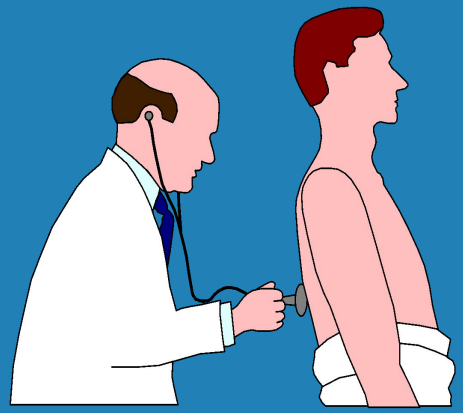


Hooke
d



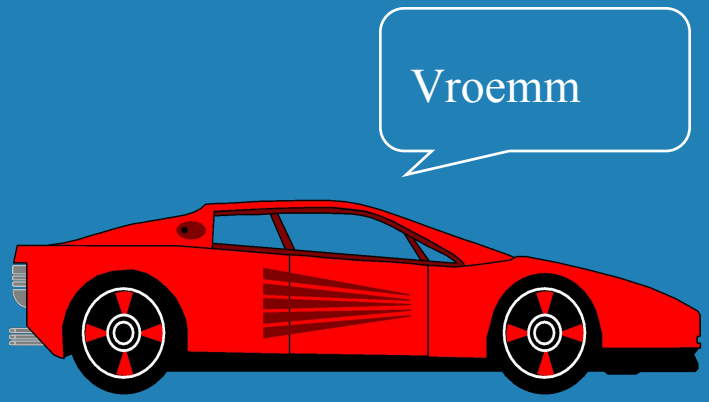
Broken

Happy



il
l

Objects have behavior....



World

- The world is
  - a set of things
  - interacting with each other.
- OOP is more natural to humans, but less natural to computers
- Computers (usually) have a single thread of control, so objects take turns

Describing the world

- **Describe a particular person**
  - **Snigdha has long blond hair, green eyes, is 1.63m tall, weighs 46Kg and studies computer engineering. Now sitting in the class room.**
  - **Raina has short black hair and brown eyes. He is 180cm and 55 kilos. Now thinking to take a nap!**
- **Notice how all have specific values of**
  - **name, height, weight, eye colour, state, …**

Object Properties

- **Identity**
- **State**
- **Behavior**

on
of
f

myLam
p

Object is an abstraction of a real world entity

Introduction to Objects

- An *object* represents something with which we can interact in a program
- An object provides a collection of services that we can tell it to perform for us
- The services are defined by methods in a *class* that defines the object
- A class represents a concept, and an object represents the embodiment of a class
- A class can be used to create multiple objects

Objects and Classes

A class (the concept)

An object (the realization)

Siam's Bank Account
Balance: $5,257

Rain's Bank Account
Balance: $1,245,069

Muhaimin's Bank Account
Balance: $16,833

Multiple objects from the same class

Java OOP terminology

- Class - **Category**
  - **Properties/states**
  - **Functionality/Services (examines/alters state)**

- **object** - Individual/unique thing
  (an instance of a class)
  - Particular value for each property/state
  - & functionality of all members of class.

Java OOP Software

- **Software System**
  - **Set of objects**
  - **Which interact with each other**

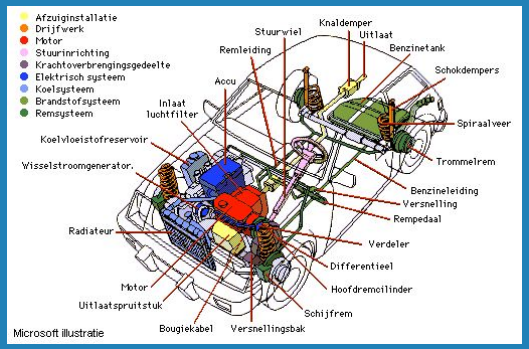
Created (instantiated) from class definitions

One object will send a message to another object asking it to do a particular task. The first object does not need to know how the task is done (only how to request that it be done.)

This corresponds to calling one of the second object's methods!

Abstraction

- An *abstraction* hides (or ignores) unnecessary details
- denotes the essential properties of an object
- One of the fundamental ways in which we handle complexity
- Objects are abstractions of real world entities
- Programming goal: choose the right abstractions



Abstraction

A car consists of four wheels an engine, accumulator and brakes.

Multiple Abstractions

A single thing can have multiple abstractions

Example: a protein is…

- **a sequence of amino acids**
- **a complicated 3D shape (a fold)**

Choosing Abstractions

Abstractions can be about

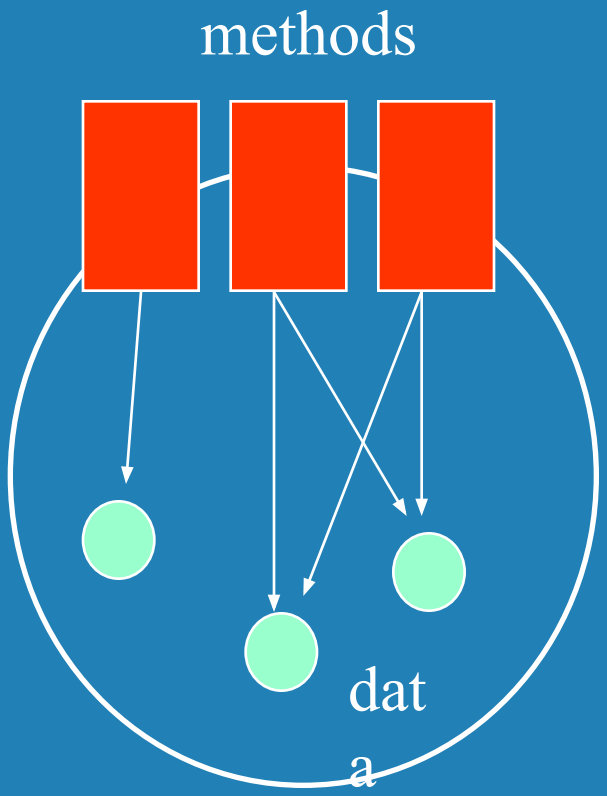
- tangible things (a vehicle, a car, a map) or
- intangible things (a meeting, a route, a schedule)

An example:

- Abstraction name: light
- Light's wattage (i.e.,energy usage)
- Light can be on or off

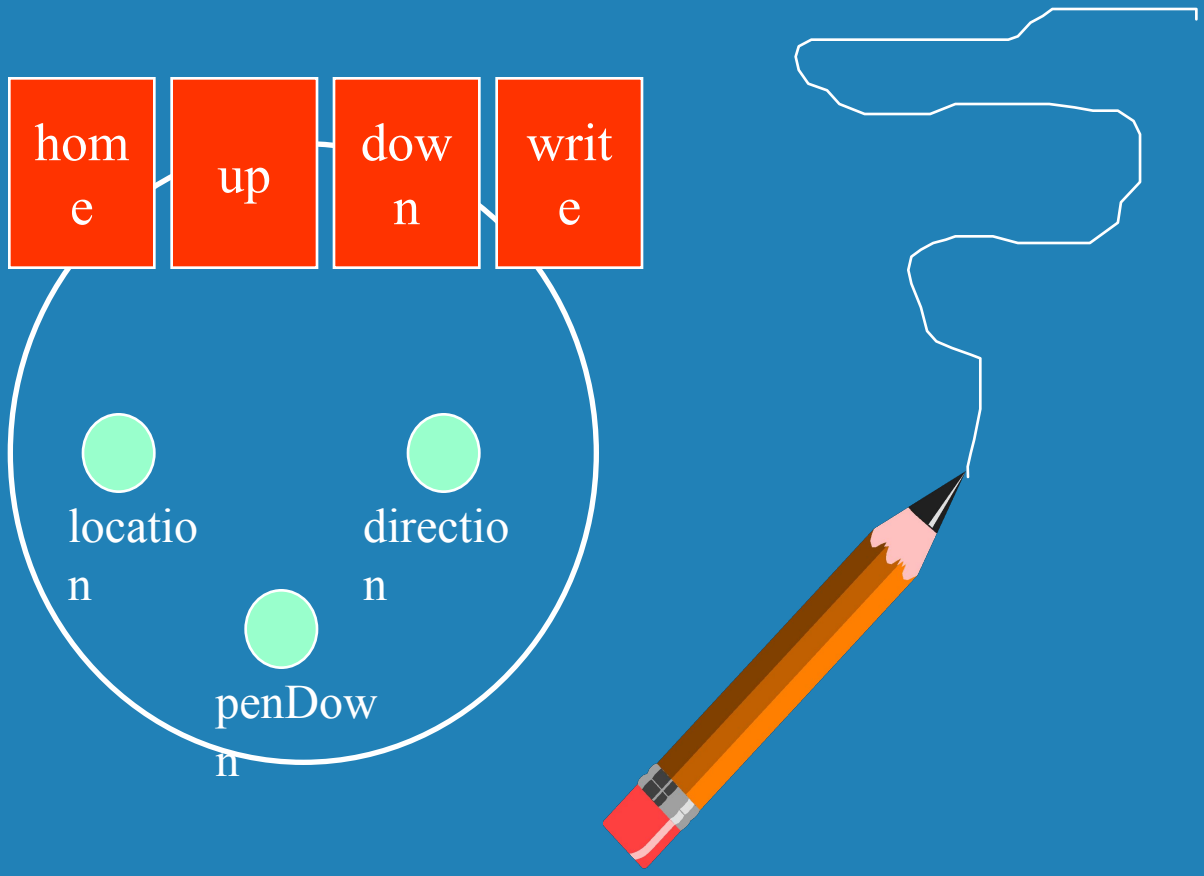
- There are other possible properties (shape, color, socket size, etc.), but we have decided those are less essential
- The essential properties are determined by the problem

Object-Oriented Model



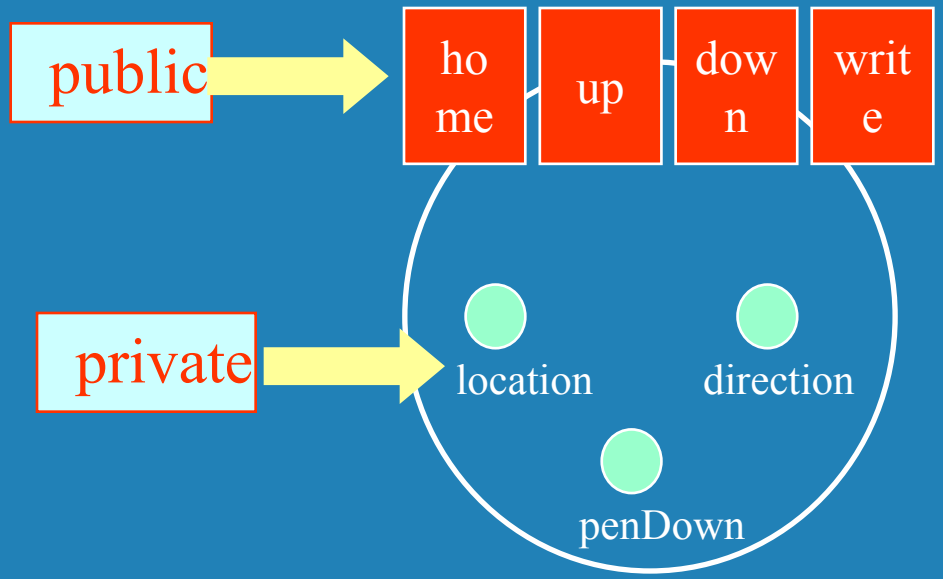
Object boundary

Example: Pencil



Encapsulation

- data belonging to an object is hidden, so variables are *private*
- methods are *public*
- we use the public methods to change or access the private data



Programming Implications

- Encapsulation makes programming easier
  - As long as the contract is the same, the client doesn't care about the implementation

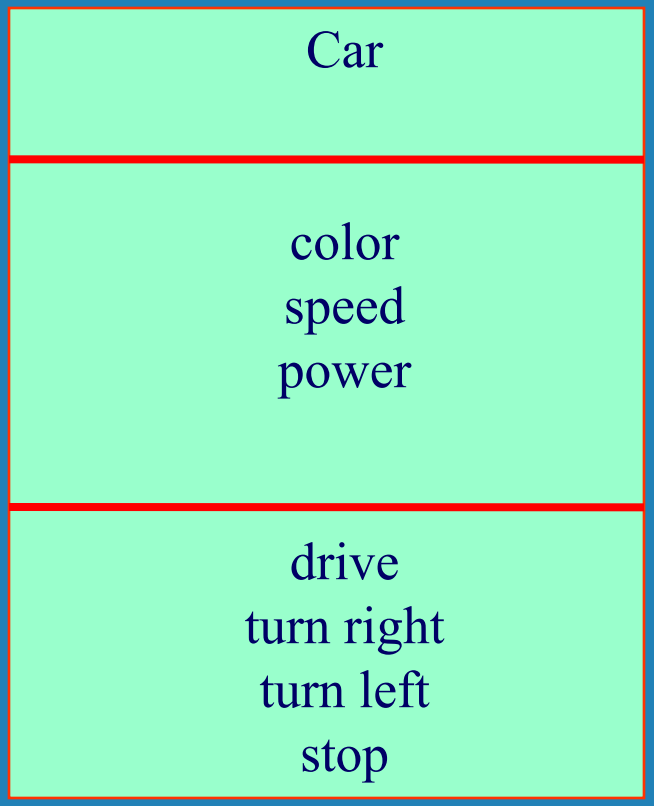
- In Java, as long as the method signatures are the same, the implementation details can be changed
  - In other words, I can write my program using simple implementations; then, if necessary, I can replace some of the simple implementations with efficient implementations

Car Objects

Defining class Car

- What are the common attributes of cars?
- What are the common behaviors of cars?

Class Car

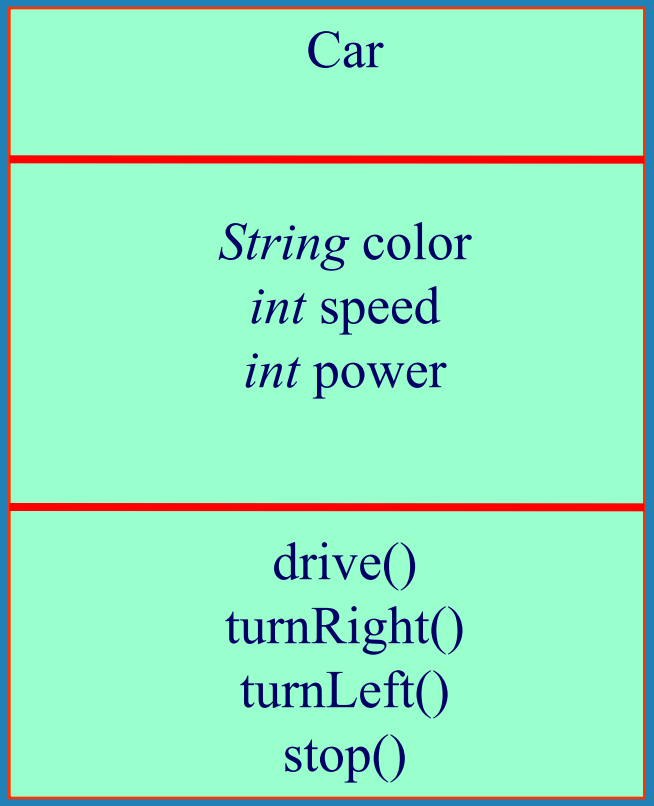


class name

attributes

operations

in Java



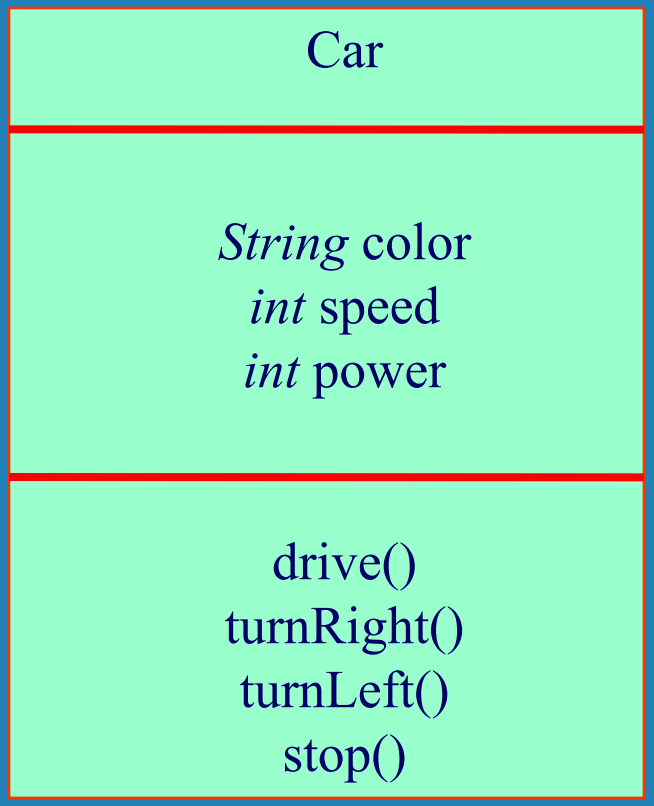
class name

attributes or
instance variables

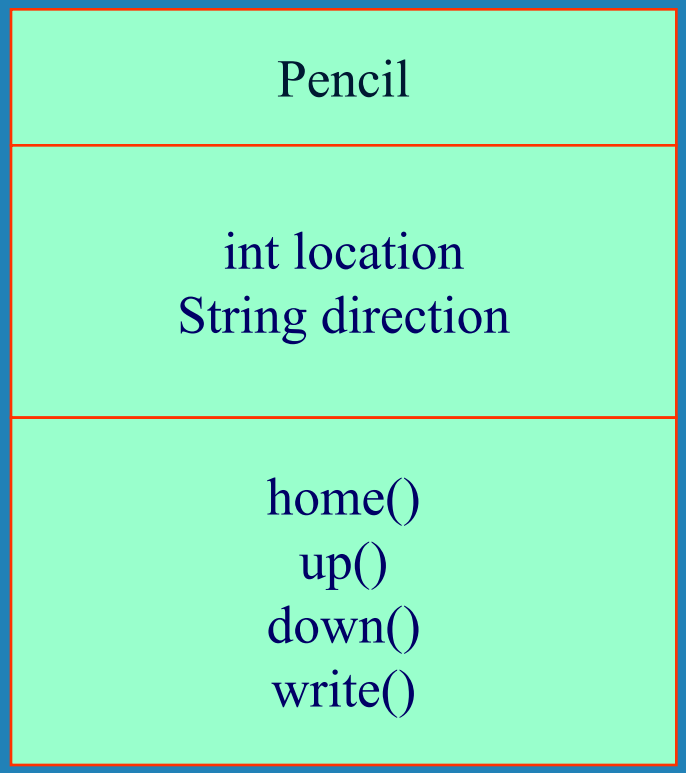
methods

Java Syntax

```
public class Car
{
// attribute declarations
        private String color;
        private int speed;
        private int power;

// method declarations
      public void drive()
      {  // ….
      }

      public void turnRight()
      {  // ….
      }

      public void turnLeft()
      {  // ….
      }

      public void stop()
      {  // ….
      }

}
```

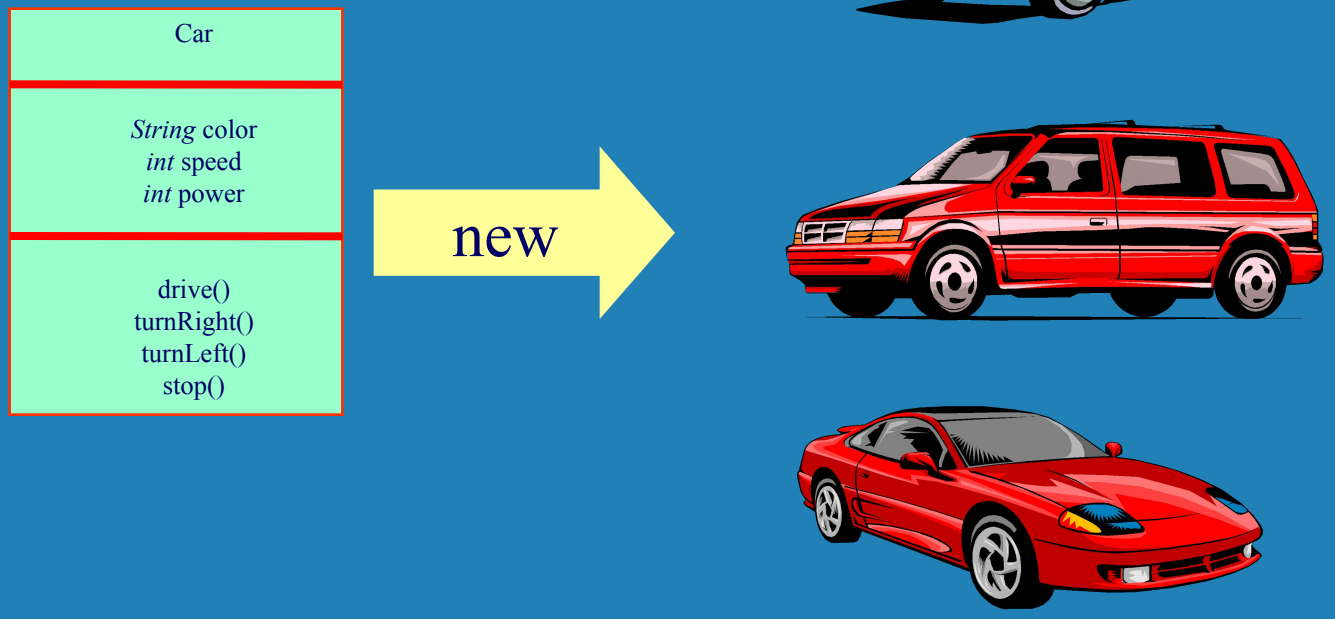


Class Pencil



Declaring objects

- A class can be used to *create* objects
- Objects are the instances of that class



Java's "Building Blocks"

- Data types
  - primitive constructs (e.g., integers, floating point numbers, characters)
- Class
  - A description of a set of objects
  - used to create objects

Primitive Data

- There are exactly eight primitive data types in Java
- Four of them represent integers:
  - `byte`, `short`, `int`, `long`
- Two of them represent floating point numbers:
  - `float`, `double`
- One of them represents characters:
  - `char`
- And one of them represents boolean values:
  - `boolean`

Declaring object variables

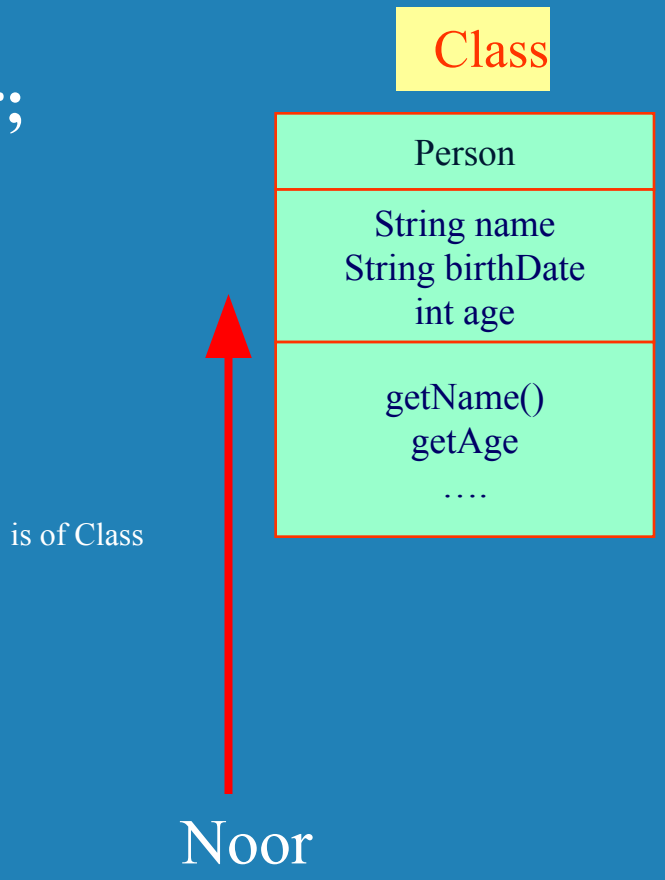
- **A class name can be used as a type to declare an *object reference variable***

```
Person Noor;
```

- **An object reference variable holds the address of an object**

Declaring Objects

Person Noor;



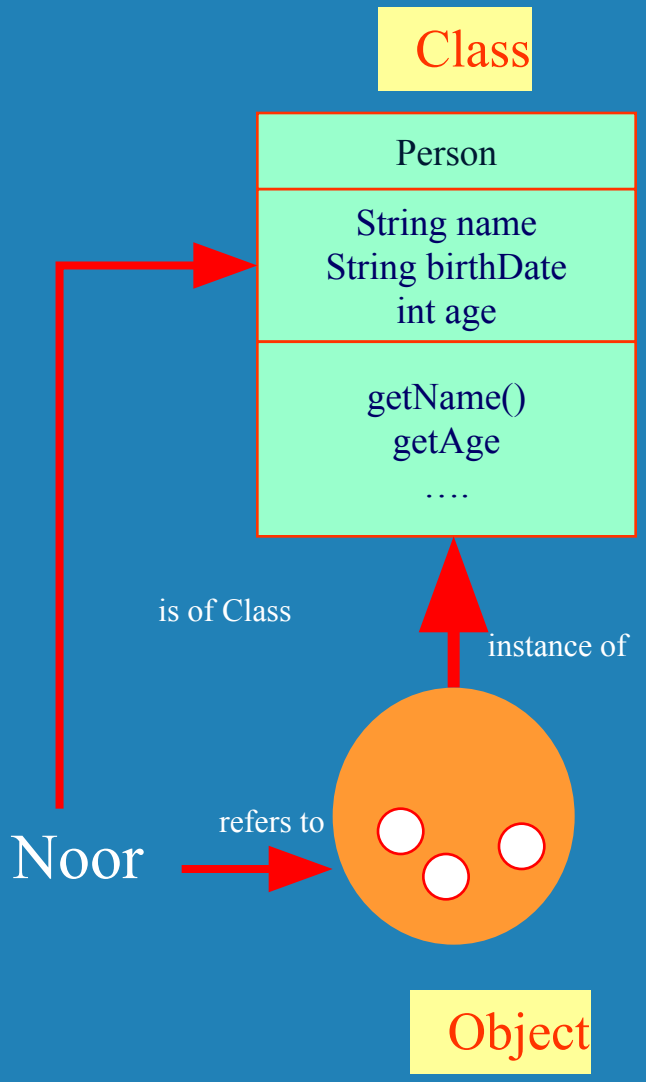
Creating Objects

- We use the `new` operator to create an object

```
Noor = new Person();
```

- Creating an object is called *instantiation*
- An object is an *instance* of a particular class
- We can combine declaration and creation:

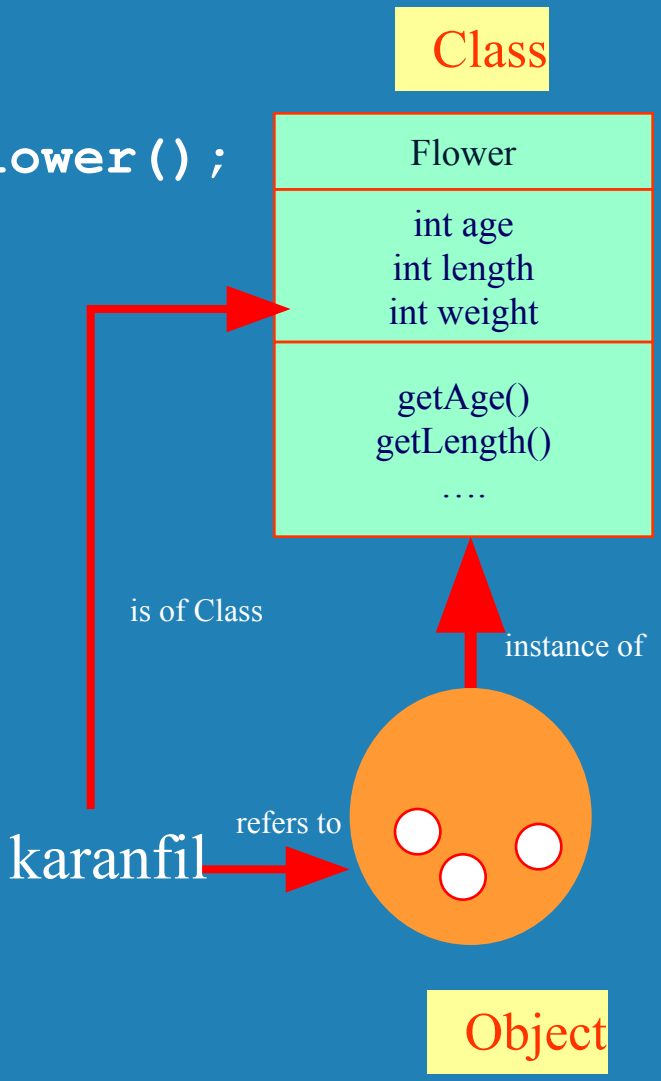
```
Person  Noor = new Person();
```



Declaring and Creating Objects

Flower karanfil;

```
karanfil = new Flower();
```



Basic approach

- Define class
- Declare objects
- Create objects
- Use objects

Using objects

- The way you work with objects is to send them messages
- Most statements using objects have the following structure

  object.method

  - for example: thisPerson.setAge(24);
- This means
  - the object whose name is thisPerson
  - is sent the message setAge()
  - along with the "value" 24
- The effect of this is to set the person's age to be 24 years old

Example

```
Person Noor;
Noor = new Person();
Noor.setName("Noor Hossain");
Noor.setAge(18);
```