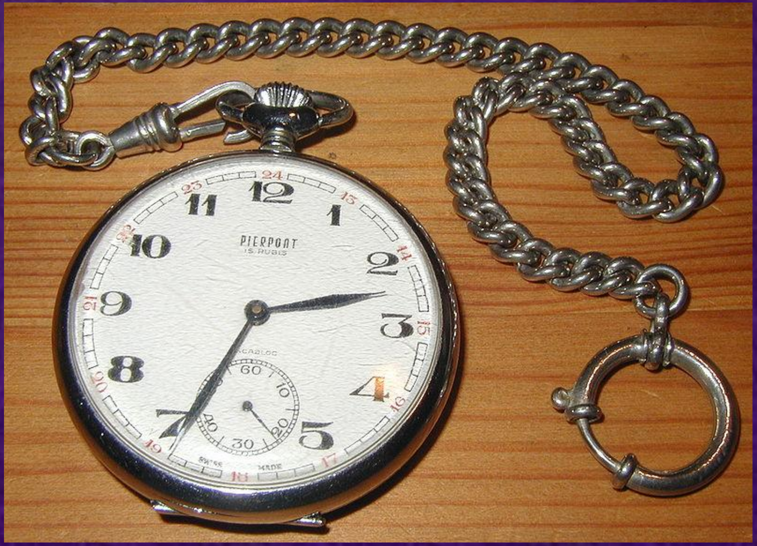
Java Technology Classes and Objects

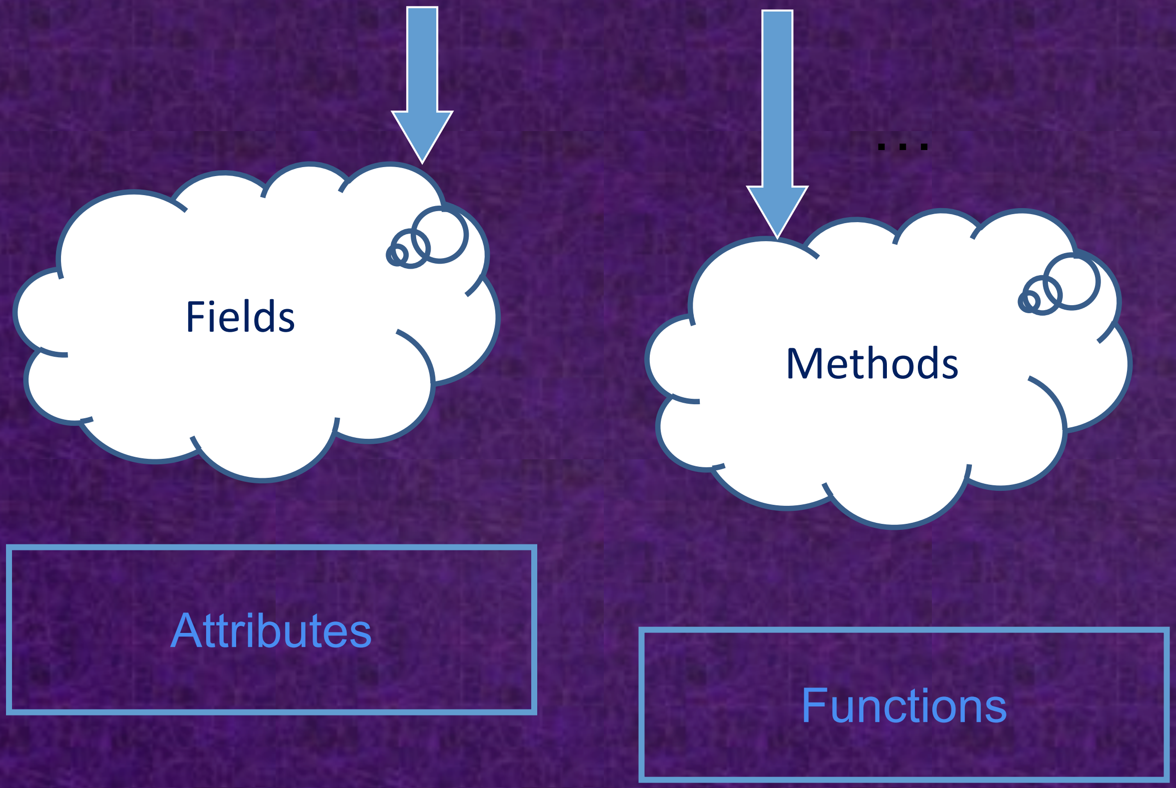
{ Dr. Sergey Krylov

- Start and finish
- Break
- Facilities, telephones and messages
- Questions and discussions

LOGISTICS



Class



<u>Class is abstact data type.</u>

Class denotes category of objects, and act as **blueprint** for creating such objects.

CLASS

Object is entity that has the following features:

- ❑ An object has **identity** (it acts as a single whole).
- ❑ An object has **state** (it has various properties, which might change).
- ❑ An object has **behavior** (it can do things and can have things done to it).

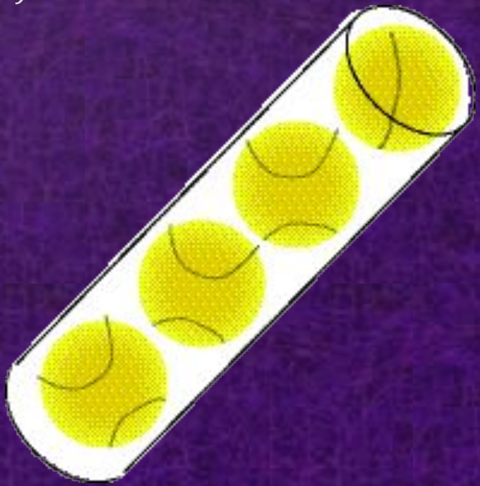
OBJECT

	Objects	**Non-objects**
1.	A pen	The upper 37 % of the pen
2.	A Computer Keyboard	The air above the keyboard
3.	A shoe	The color black
4.	A desk	All desks in the world

EXAMPLES

Consider a tube of four yellow tennis balls.

- Is the tube of tennis balls an object?
- Is each tennis ball an object?
- Could the top two balls be considered a single object?
- Is the color of the balls an object?
- Is your understanding of tennis balls an object?



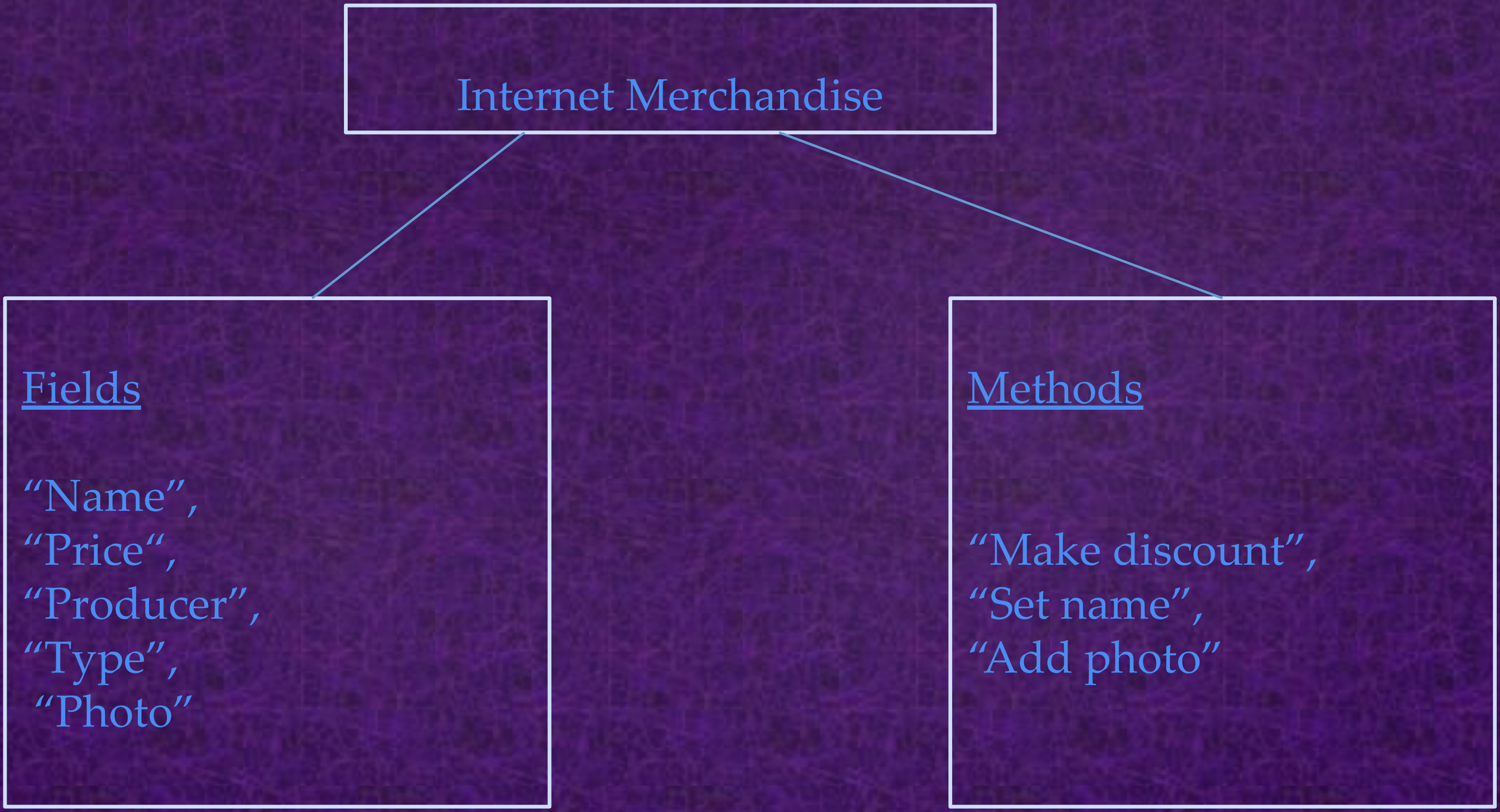
EXERCISE

- Question: What are software objects made out of?
- Answer: Computer memory.
- Software objects have **identity** because each is a separate chunk of memory having *name*.
- Software objects have **state.** Some of the memory that makes a software object is used for *attributes* which contain values.
- Software objects have **behavior.** Some of the memory that makes a software object is used to contain programs (called *methods*) that enable the object to "do things." The object does something when one of its method runs.

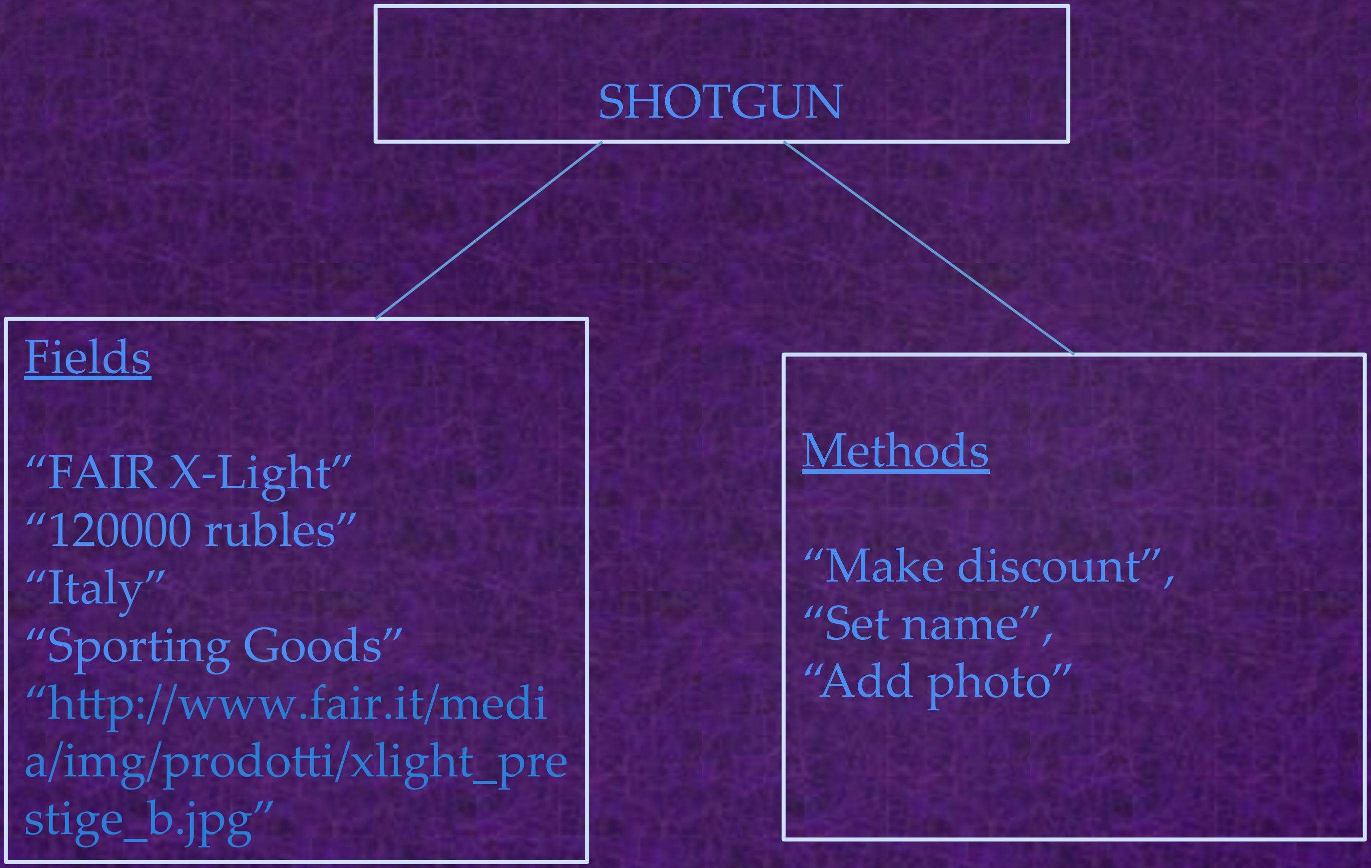
SOFTWARE OBJECTS

- A **class** is a description of a kind of object.
  - A programmer may define a class
  - or may use predefined classes that come in class libraries.
- A class is merely a plan for a possible object (or objects.). It does not by itself create any objects.
- When a programmer wants to create an object the new operator is used with the name of the class.
- Creating an object is called **instantiation**.

SOFTWARE CLASS



CLASS EXAMPLE



- Object of same class have <u>same structure</u>, it exhibits properties & behaviors defined by its class.
- Properties is also called as attributes/fields & behaviors as methods/operations.

OBJECT EXAMPLE

```
class <class_name>{
    field;
    method;
}
```

A class in java can contain:

- field
- method
- constructor
- block
- class and interface

CLASS IN JAVA

Write class Vehicle that have attributes (fields) speed и power.

```
public class Vehicle{
int speed;
double power;
}
```

Declare objects auto and moto.

```
Vehicle auto;
Vehicle moto = null;
```

Create objects auto and moto by **new** keyword.

```
auto = new Vehicle();
moto = new Vehicle();
```

EXAMPLE

Code that is outside the object's class must use an object reference or expression, followed by the dot (.) operator, followed by a simple field name, as in:

objectReference.fieldName

```
To get:    variable = object.field;
To set:    object.field = variable;
```

GET & SET ATTRIBUTES

Write program that sets different values for the fields:

```
auto.power = 88.7;
auto.speed = 150;
moto.power = 24;
moto.speed = 120;
System.out.println(auto.power +” “+ auto.speed);
System.out.println(moto.power +” “+ moto.speed);
auto.speed = 180;
System.out.println(auto.power +” “+ auto.speed);
System.out.println(moto.power +” “+ moto.speed);
```

Every object has independent fields.

EXERCISE

CALLING METHODS

Use an object reference to invoke an object's method. Append the method's simple name to the object reference, with an intervening dot operator (.). Provide, within enclosing parentheses, any arguments to the method. If the method does not require any arguments, use empty parentheses.

objectReference.methodName(argumentList);

or:

objectReference.methodName();

Add method to calculate effectiveness into Vehicle class

```
double eff() {
return power/speed;
      }
```

Compare 2 objects

```
System.out.print(auto.eff()>moto.eff());
```

The objects must be created to use method eff().

EXERCISE

- **String is a very special class in Java.**
- Strings are **constants**, their values cannot be changed after they are created. But they can be reinitialized.

String here="I am here";

here="I am there";//previous one is completely deleted

- Our rules and dictionaries can be stored string class.
- This class is armed with useful methods

compareTo(String str),
concate(String str),
endsWith(String str),
indexOf(int ch),
length(),
startsWith(String str),
lastIndexOf(String str).

CLASS STRING

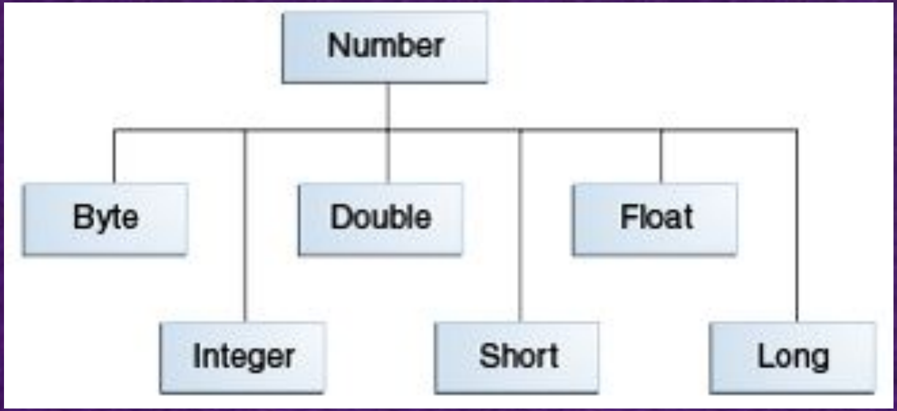
```
String machine = “pentium”;
int comp = machine.compareTo(“pentium”); //comp=0;
String lab= “Language”;
lab=lab.concate(“ Technology”); //lab=“Language Technology”;
int ind = machine.indexOf(‘t’);
boolean start = machine.startsWith(“pen”); //true
boolean end = machine.endsWith(“um”); //true
String part1=machine.substring(0,3); //part1=“pen”;
String part2=machine.substring(5);//part2=“um”;
```

STRING METHODS

```
class stringTester {
      public static void main ( String[] args ) {
        String str1;         // str1 is a variable that refers to an object,
                        // but the object does not exist yet.
        int len; // len is a primitive variable of type int
        str1 = new String(“Bullet is waiting…"); // create an object of type String
        len = str1.length(); // invoke the object's method length()
      System.out.println("The string is " + len + " characters long");
      }
 }
```

EXAMPLE

Java API provides *wrapper* classes for each of the primitive data types. These classes "wrap" the primitive in an object.



```
int x = 25;
Integer y = new Integer(33);
```

NUMBER WRAPPERS

There are three reasons that you might use a Number object rather than a primitive:

- As an argument of a method that expects an object (often used when manipulating collections of numbers).
- To use constants defined by the class, such as MIN_VALUE and MAX_VALUE, that provide the upper and lower bounds of the data type.
- To use class methods for converting values to and from other primitive types, for converting to and from strings, and for converting between number systems (decimal, octal, hexadecimal, binary).

REASONS

To convert the value of this Number object to the primitive data type returned.

```
byte byteValue()
short shortValue()
int intValue()
long longValue()
float floatValue()
double doubleValue()
```

```
Double Var = new Double (3.1415);
int i = Var.intValue();
```

CONVERT TO PRIMITIVE TYPES

To compare this Number object to the argument.

```
int compareTo(Byte anotherByte)
int compareTo(Double anotherDouble)
int compareTo(Float anotherFloat)
int compareTo(Integer anotherInteger)
int compareTo(Long anotherLong)
int compareTo(Short anotherShort)
```

```
Integer Var = new Integer (5);
int i = 3;
if(Var.compareTo(i)) System.out.println (“They are the same!”);
else System.out.println  (“They are different!”);
```

COMPARATION

Each Number class contains other methods that are useful for converting numbers to and from strings and for converting between number systems.

static Integer decode(String s)	Decodes a string into an integer. Can accept string representations of decimal, octal, or hexadecimal numbers as input.

```
Integer iVar = Integer.decode(“3”);
```

static int parseInt(String s)	Returns an integer (decimal only).

```
int i = Integer.parseInt(“3”);
```

static int parseInt(String s, int radix)	Returns an integer, given a string representation of decimal, binary, octal, or hexadecimal (radix equals 10, 2, 8, or 16 respectively) numbers as input.

```
int i = Integer.parseInt(“3”,16);
```

STRING INTO INTEGER

String toString()

```
Integer y = 125;
String s1 = y.toString(); //"125”
```

Returns a String object representing the value of this Integer.

static String toString(int i)

Returns a String object representing the specified integer.

```
String s1 = Integer.toString(25); //"25”
```

static String **toBinaryString**(int i)
public static String **toOctalString**(int i)
public static String **toHexString**(int i)

Returns a String object representing the specified integer in binary, octal or hexadecimal form.

```
String s1 = Integer.toBinaryString(12); //“1100”
```

INTEGER TO STRING

Write program which gets input of the user in decimal integer form and which prints the result in binary, octal and hexadecimal form.

EXERCISE

Write class to represent Radio that have name, frequency and mode (switched on/off). Have methods to change the mode and to change frequency. In main() method create Radio object and then get user input to change and to print mode & frequency.

HOMEWORK

QUESTIONS?