INHERITANCE

One way that we humans make sense of the surrounding physical world is by categorizing the things around us into classes and organizing those classes according to common characteristics.

Example
Eight objects are pictured below, all of which you can immediately recognize as musical instruments. All eight objects produce sound – the two drums by percussion; the guitar, banjo and harp by strings; and the three horns by blowing air into them.

IS-A
Objects are often related through specialization and generalization. If object \( A \) is more specific than object \( B \), then \( A \) is a specialization of \( B \) and \( B \) is a generalization of \( A \). Computer programmers call this the IS-A relationship and say that \( A \) IS-A \( B \).

Example
A police car IS-A car; it is a specialization of a car. Like all cars, it has four wheels, an engine, etc. In addition, it has attributes that an ordinary car doesn’t have, such as a siren and flashing lights.
In Java, to declare a new class to be a specialization of an existing (i.e. previously declared) class, use the syntax:

```
class new class name extends existing class name
```

The new class is the subclass; the existing class is the superclass.

**Example**

Given the following Java declarations that model pens and pencils.

```
A Pen IS-A WritingImplement
A Pencil IS-A WritingImplement

Pen and Pencil are the subclasses
WritingImplement is the superclass
```

```
1  public class WritingImplement
2  {
3      public Color color;
4      . . .
5  }
6
7  public class Pen extends WritingImplement
8  {
9      public double length;
10     . . .
11  }
12
13  public class Pencil extends WritingImplement
14  {
15     public boolean hasEraser;
16     . . .
17  }
```

Java implements the concept of specialization with a mechanism called *inheritance*. A subclass automatically inherits all of the fields and methods of its superclass. The subclass can have additional fields and methods that the superclass does not have.

**Example**

Pencil has two fields – color and hasEraser. Pen has two fields – color and length.
**Class Diagrams**

The **IS-A** relationship between classes is often represented in a picture called a *class diagram*. The arrows in the diagram point from the subclass to the superclass.

*Example*

This class diagram pictures **WritingImplement** with its two subclasses.

![Class Diagram](image)

Subclasses can themselves be extended into other subclasses.

*Example*

Subclasses of **Pencil**.

```java
1 public class MechanicalPencil extends Pencil
2 {
3     public double leadWidth;
4         ...
5 }        
6 7 public class WoodenPencil extends Pencil
8 {
9     public int grade;
10             ...
11 }
```

A superclass and its descendant subclasses form a *class hierarchy*.

*Example*

![Class Diagram](image)
**java.lang.Object**

Any class declared in a Java program extends, by default, class `java.lang.Object` in the Java API. In other words, all classes in Java are organized into a gigantic class hierarchy with `Object` as the ultimate superclass.

**Example**

The `WritingImplement` hierarchy in context with `Object`.

```
Object
   ^ IS-A
WritingImplement
   |    color: Color
Pencil
   |    color: Color
   |    hasEraser: boolean
   v IS-A
WoodenPencil
   |    color: Color
   |    hasEraser: boolean
   |    grade: int
   v IS-A
MechanicalPencil
   |    color: Color
   |    hasEraser: boolean
   |    leadWidth: double
Pen
   |    color: Color
   |    length: double
```
