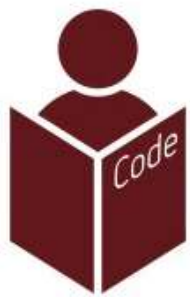
Java Extras

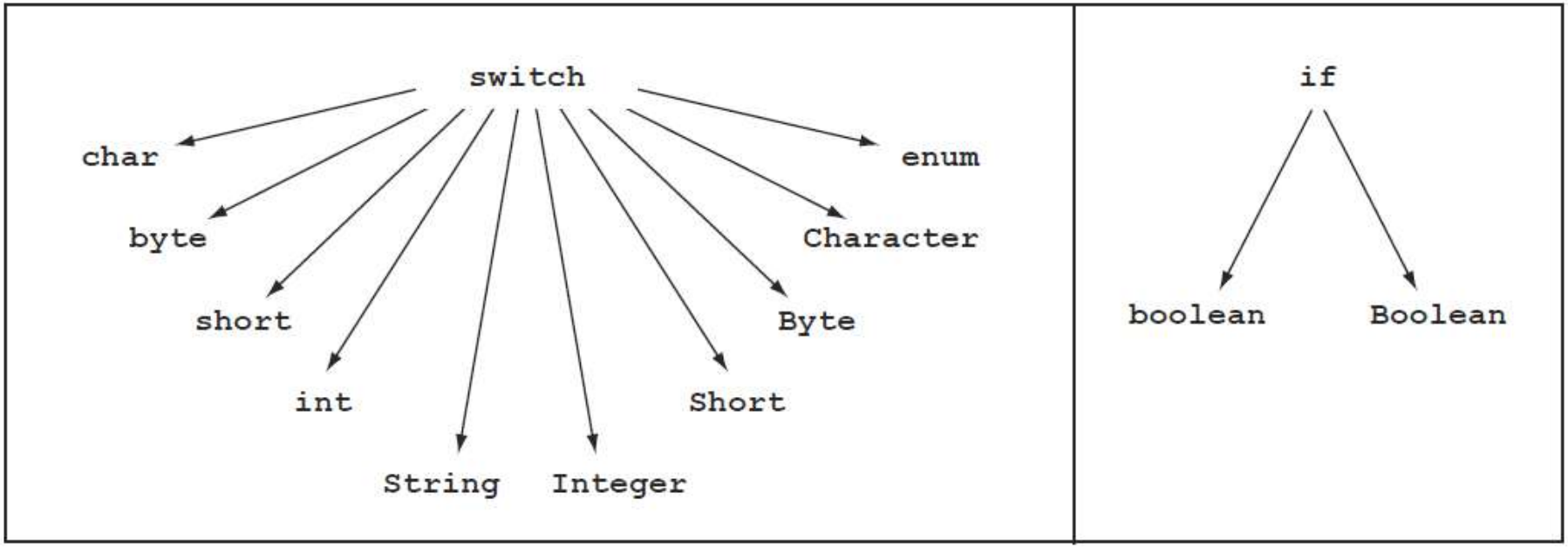
Ternary Operator ?:



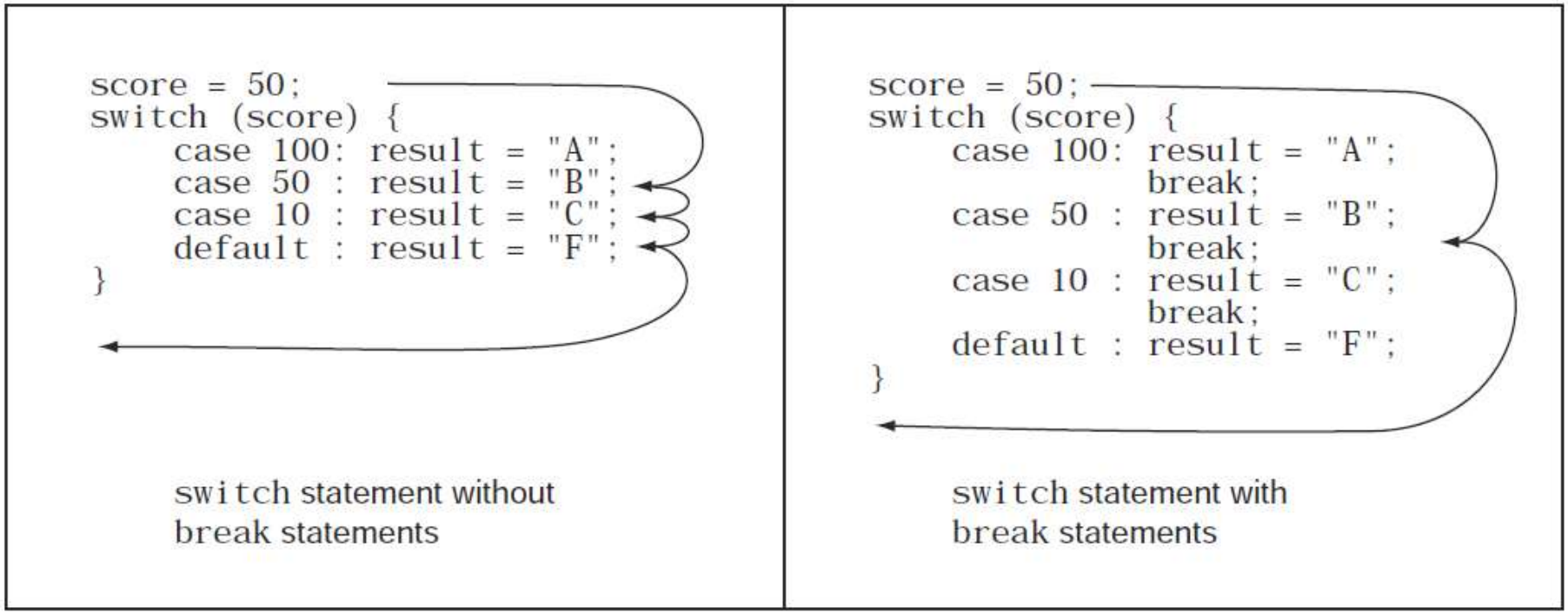
Ternary construct	if-else construct

`int bill = 2000;`
`int discount = (bill > 2000)? 15 : 10;`
	`int bill = 2000;`
	`int discount`
	`if (bill > 2000)`
	`    discount = 15;`
	`else`
	`    discount = 10;`

Switch Vs If


switch
char
byte
short
int
String
Integer
Short
Byte
Character
enum
if
boolean
Boolean

Προσοχή με τη χρήση Switch



Προαιρετικά πεδία στο for loop

```
int a = 10;
for(; a < 5; ++a) {
    System.out.println(a);
}
```

Valid for loop without any code in the initialization block

```
for(int a = 10; ; ++a) {
    System.out.println(a);
}
```

Missing termination condition implies infinite loop

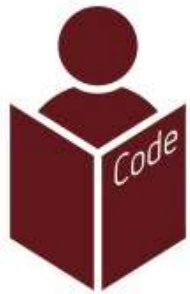
```
for(int a = 10; a > 5; ) {
    System.out.println(a);
}
```

Missing update clause

Enhanced for loop (*for-each*)


```
int total = 0;
int primeNums[] = {2, 3, 7, 11};
for (int num : primeNums)
    total += num;
```

for Vs for-each



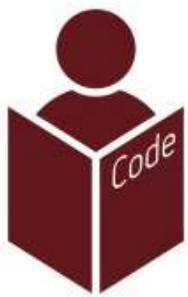
- The enhanced `for` loop can't be used to initialize an array and modify its elements.
- The enhanced `for` loop can't be used to delete the elements of a collection.
- The enhanced `for` loop can't be used to iterate over multiple collections or arrays in the same loop.

Java – Labeled Statements

- Στη Java μπορούμε να χρησιμοποιήσουμε labels στις παρακάτω περιπτώσεις τμημάτων κώδικα:

  - A code block defined using { }
  - All looping statements (`for`, enhanced `for`, `while`, `do-while`)
  - Conditional constructs (`if` and `switch` statements)
  - Expressions
  - Assignments
  - `return` statements
  - `try` blocks
  - `throws` statements

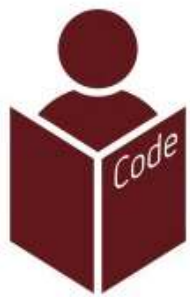
Labeled break




```
String[] programmers = {"Outer", "Inner"};
outer:
for (String outer : programmers) {
    for (String inner : programmers) {
        if (inner.equals("Inner"))
            break outer;
        System.out.print(inner + ":");
    }
}
```

Exits the outer loop, marked with label outer

Labeled continue




```
String[] programmers = {"Paul", "Shreya", "Selvan", "Harry"};
outer:
for (String name1 : programmers) {
    for (String name : programmers) {
        if (name.equals("Shreya"))
            continue outer;
        System.out.println(name);
    }
}
```

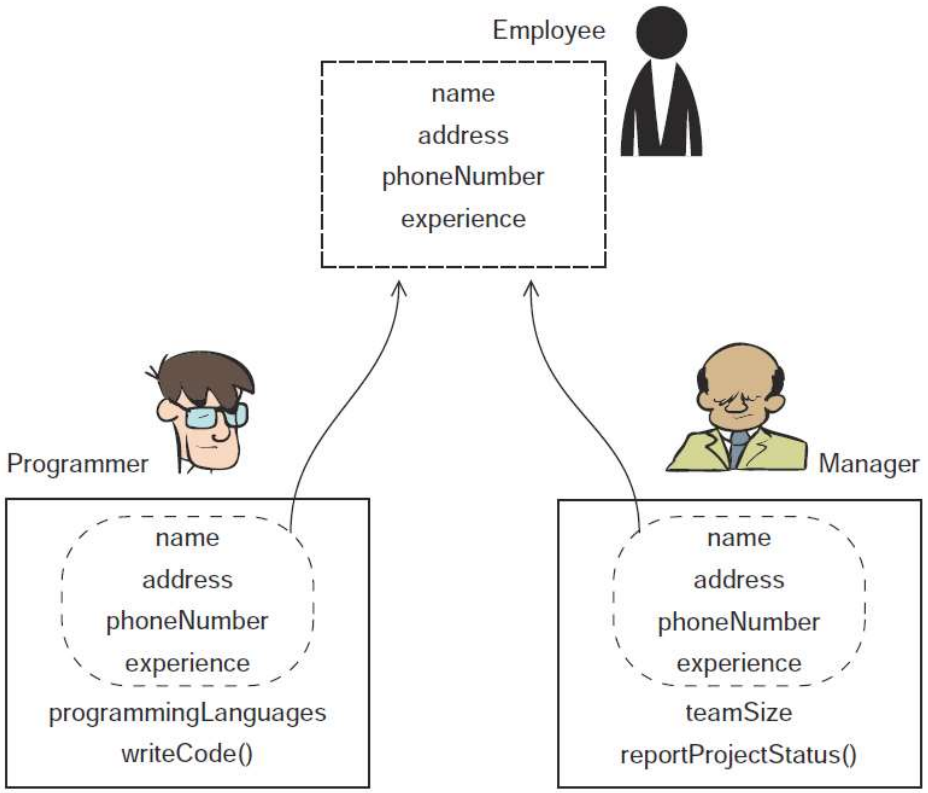
Skips remaining code for current iteration of outer loop and starts with its next iteration

Σύνοψη – Branching statements

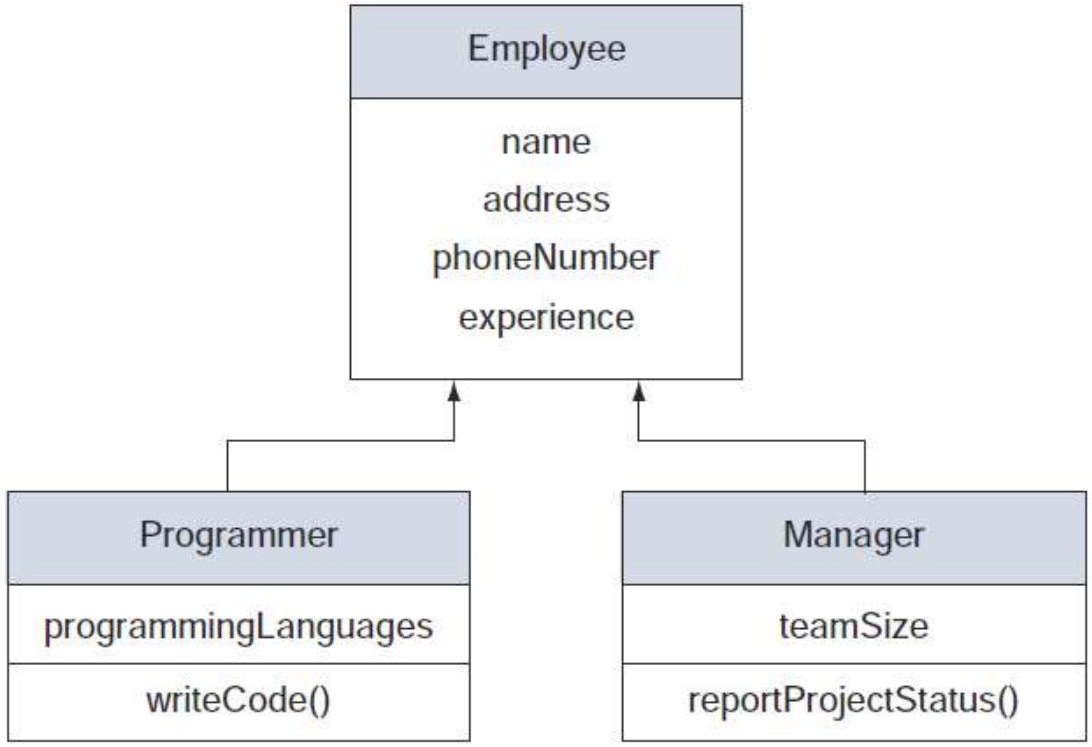
- Break
  - Unlabeled
  - Labeled
- Continue
  - Unlabeled
  - Labeled
- Return
  - With value
  - No value

Java Inheritance

Inheritance 1/3



Inheritance 2/3



Inheritance 3/3


```
class Employee {
    String name;
    String address;
    String phoneNumber;
    float experience;
}
class Programmer extends Employee {
    String[] programmingLanguages;
    void writeCode() {}
}
class Manager extends Employee {
    int teamSize;
    void reportProjectStatus() {}
}
```

Inheritance Σημειώσεις

- In the Java language, classes can be derived from other classes, thereby inheriting fields and methods from those classes
- A class that is derived from another class is called a subclass
- The class from which the subclass is derived is called a superclass

What can a Subclass do?

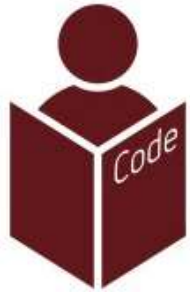
- The inherited fields can be used directly, just like any other fields
- You can declare new fields in the subclass that are not in the superclass
- The inherited methods can be used directly as they are
- You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it
- You can declare new methods in the subclass that are not in the superclass

Inheritance Example

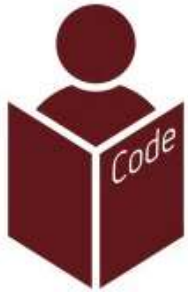
```
public class Animal{
}

public class Mammal extends Animal{
}

public class Reptile extends Animal{
}

public class Dog extends Mammal{
}
```

IS-A vs HAS-A

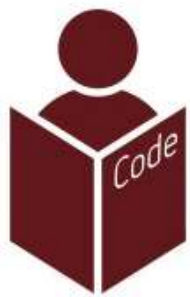
- IS-A σημαίνει «είναι του τύπου» και υποδηλώνει κληρονομικότητα (Inheritance) μεταξύ αντικειμένων
- HAS-A σημαίνει «έχει» ή «διαθέτει» και υποδηλώνει σχέση σύνθεσης (Composition) μεταξύ αντικειμένων




```java
package com.unipi.talepis;

class Car {
    // Methods implementation and class/Instance members
    private String color;
    private int maxSpeed;
    void carInfo(){
        System.out.println("Car Color= "+color + " Max Speed= " + maxSpeed);
    }
    void setColor(String color) {
        this.color = color;
    }
    void setMaxSpeed(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }
}
```

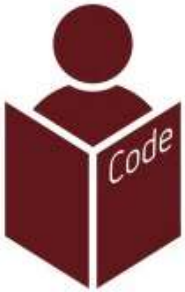



```java
package com.unipi.talepis;

public class Engine {
    void start(){
        System.out.println("Engine Started:");
    }
    public void stop(){
        System.out.println("Engine Stopped:");
    }
}
```




```java
package com.unipi.talepis;

class Beetle extends Car{
    //Beetle extends Car and thus inherits all methods
    //from Car (except final and static)
    //Beetle can also define all its specific functionality
    void BeetleStartDemo(){
        Engine beetleEngine = new Engine();
        beetleEngine.start();
    }
}
```

```java
package com.unipi.talepis;

public class Main {

    public static void main(String[] args) {
        Beetle myBeetle = new Beetle();
        myBeetle.setColor("Blue");
        myBeetle.setMaxSpeed(200);
        myBeetle.carInfo();
        myBeetle.BeetleStartDemo();
    }
}
```



Main

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" "-javaage
Car Color= Blue Max Speed= 200
Engine Started:

Process finished with exit code 0
```

Class Diagram