**LESSON 7**

1. Default Method :

* is a fully implemented method within an interface, whose declaration begins with the keyword default.
* eliminate the need to create special classes that represent a default implementation of the interface

1. Static Method:

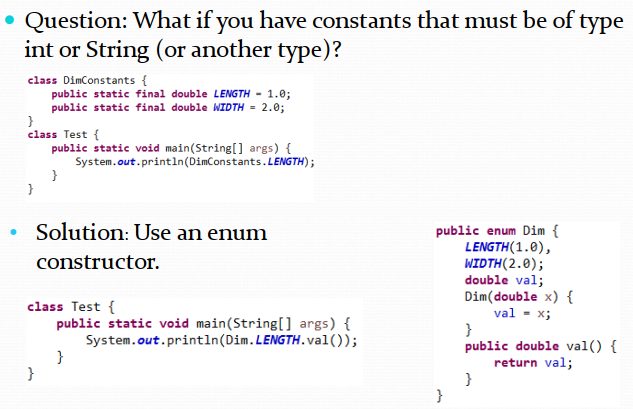
* is a fully implemented static method having the same characteristics as any static method in a class
* eliminate the need to create special utility classes that naturally belong with the interface

1. Solution to evolving API problem

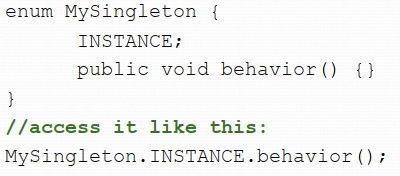
* When we add a new method to new interface with default implementation:
  + Legacy code won’t need to be required to implement new default methods, so existing code won’t be break
  + New functionality will be available for new client projects

1. application of default method:

* Enums can inherit from other type which is interface
* Using enums as constants in an application
* Optimal, threadsafe implementation of the Singleton Pattern
* Enum is not allowed to inherit from any other class
* Enum can implement interface



* Enum implementation for singleton class

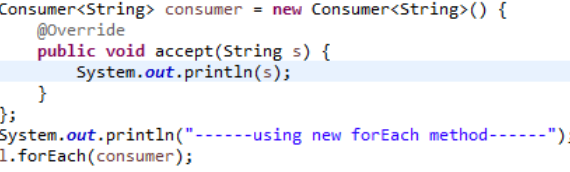


1. Iterator:

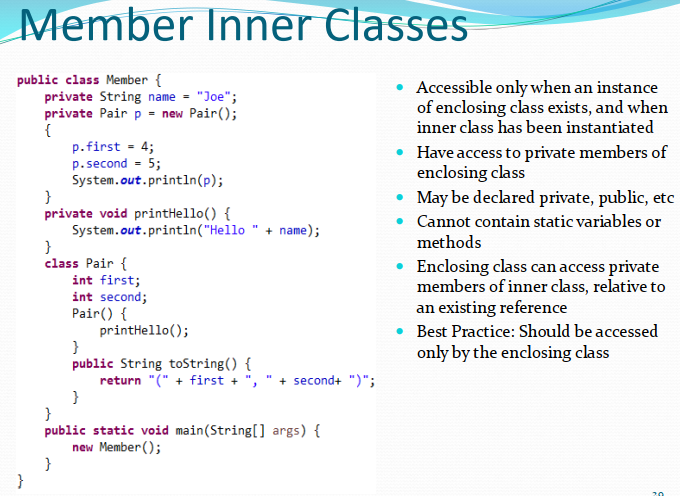
* Iterable interface supports iterate through a collection
* The only method in Iterable is iterator() -> return an iterator
* Iterator has two methods : hasnext(), next()

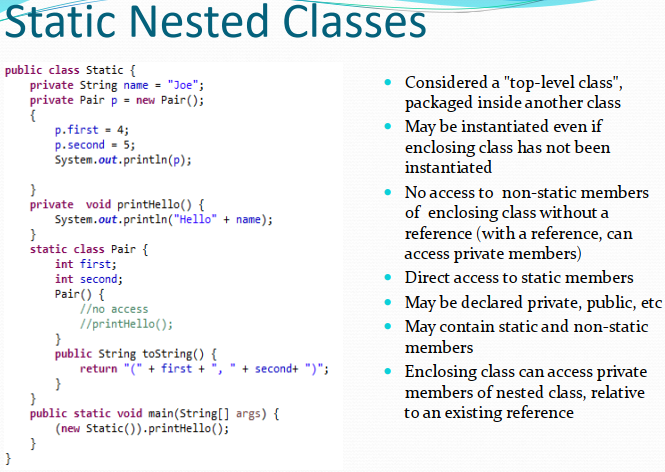
1. Consumer:

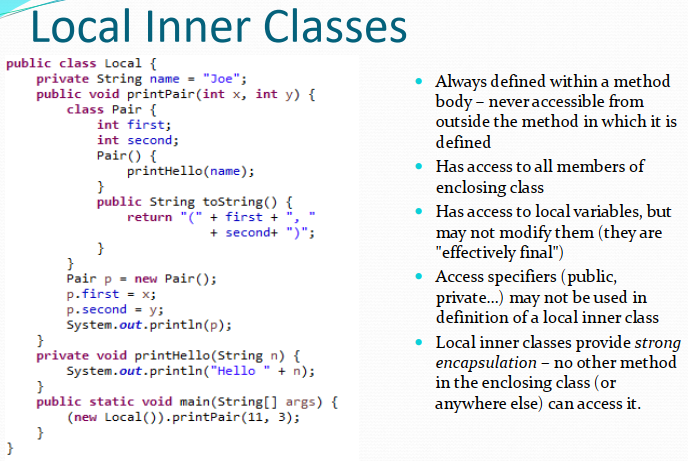
* Is a new interface in Java 8 with one abstract method accept a single argument that no return value

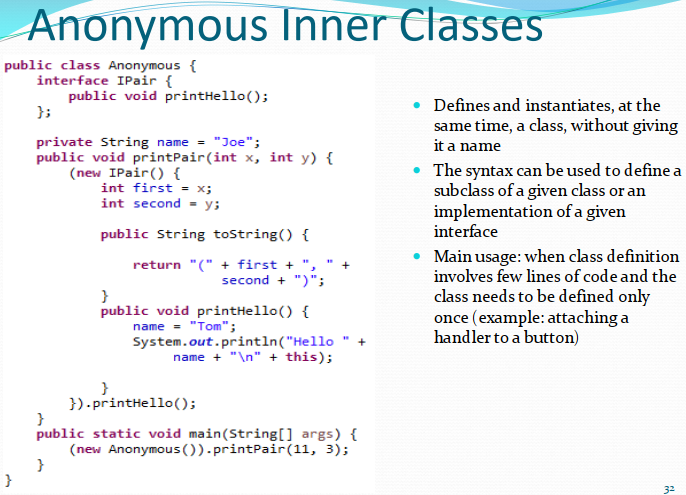


1. Nested Classes





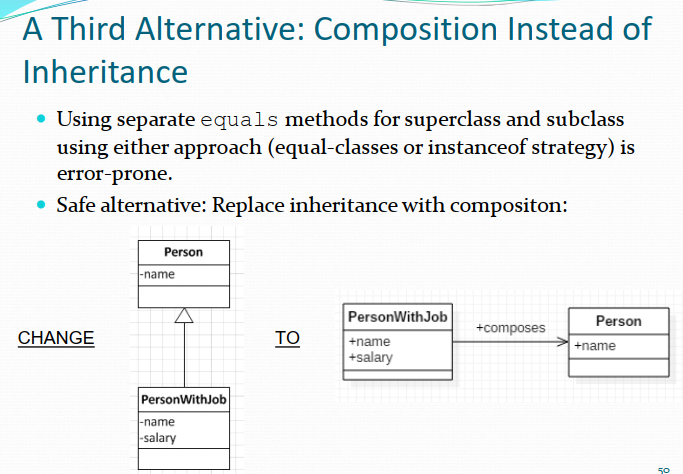


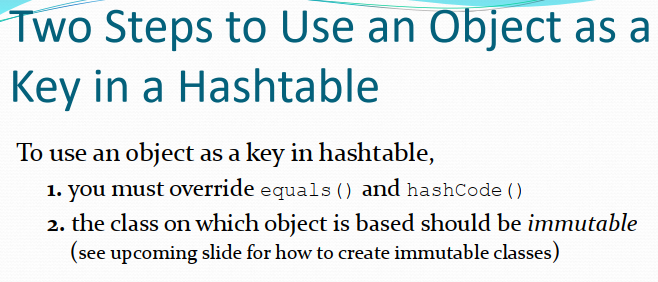


1. Default method rules in Interface

* If a class implements an interface with default method, that class inherit the default method or can override it
* Potential clash if :
  + Two interface has the same default method
  + Interface and super class has the same default method
* Static method doesn’t clash if two interface has the same name and implementation inside

1. Object has 3 base methods : equal(Object e), hashcode,toString





1. A class is immutable if the data it stores cannot be modified once it is initialized

* All fields should be private and final
* Return getter not setter
* Mark class final to not be inherited
* Mark getters don’t return mutable object

**LESSON 8**

1. Programs are declarative (“what”) rather than imperative (“how”)
2. Side effect free function:

* Do not change object state
* Have referential transparency
* No side effect

1. Lamda expression is a block of code with a list of formal parameters and a body.
2. Functional interface is an interface that specify exactly one abstract method
3. Anonymous way:

Collections.sort(obj, new Comparator<Person>(){

@override

Public int compare(Person p1, Person p2){

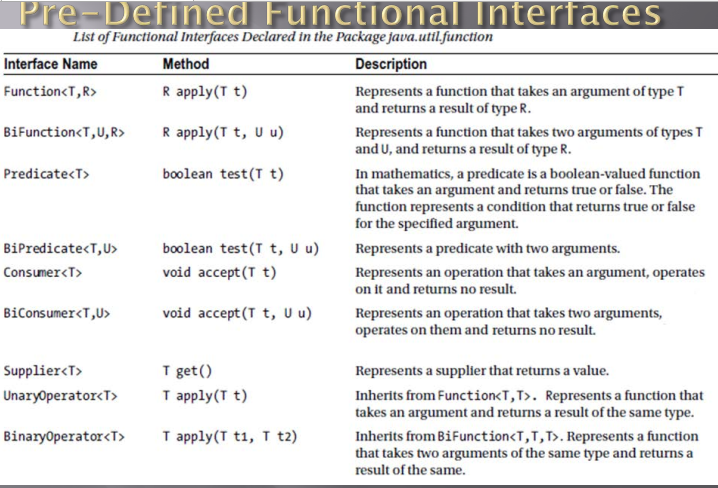
Return p1.getName().compareTo(p2.getName());

}

};)

Using Lamda:

Collections.sort(obj, (p1,p2) -> p1.getName().compareTo(p2.getName()));

1. 
2. Consumer Interface :

Consumer<String> consumer = **new** Consumer<String>() {

**public** **void** accept(String s) {

System.***out***.println(s);

}

};

Consumer<String> consumer1 = System.***out***::println;

Consumer<String> consumer2 = x -> System.***out***.println(x);

1. Predicate check is number

Predicate<String> numberOnly = x -> {

**if**(x == **null**) **return** **false**;

**return** x.chars().allMatch(Character::*isDigit*);

};

System.out.println(numberOnly.test(“abc”));

1. Free Parameter is parameter that is not defined inside a block of code
2. Closure is a block of code on the right hand side of lamda expression
3. Comparator Interface

**class** AccountComparator **implements** Comparator<Account> {

**public** **int** compare(Account a1,Account a2) {

**return** a1.getName().compareTo(a2.getName());

}

}

Comparator<Account> comparator1 = (a1,a2) -> a1.getName().compareTo(a2.getName());

Collections.*sort*(**new** ArrayList<Account>(), **new** AccountComparator());

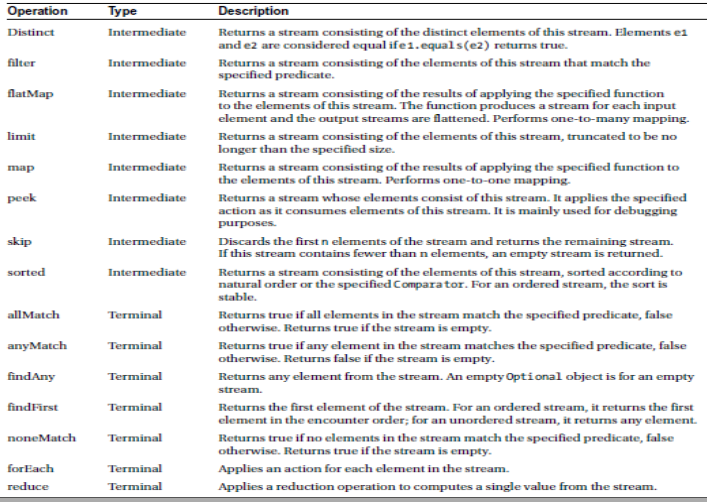
1. A method reference is a shorthand to create lamda expression using existing method : **classname::method**

**Lesson 9**

1. Monad

* Is a special data structure, available in some languages, that serves as a wrapper class, to support various operations.
* Support chaining operations, so that the output of each monad operation is another monad

1. Intermediate and Terminal Operation



1. Distinct and sorted are two statefull

