|  |  |
| --- | --- |
| Chapter 5 | |
| **Lambda**: | A lambda is a segment of code that can be stored and executed later., |
| **Stream:** | Streams unify the processing of elements of a collection and other assets of data. A stream provides useful methods to manipulate these data set. |
| *- Filter* | - Filters out a certain set of data. |
| *- Map* | - Resort the elements of a list under a new type. |
| *- Reduce* | - Reduce the list to a single element. |
| **Pipeline:** | A pipeline is the combination of two or more stream functions in a chain, where each function is applied in turns. |
|  | |

|  |  |
| --- | --- |
| Chapter 10 | |
| **Inheritance:** | Inheritance allows us to define one class as an a extension of another. |
| **Substitution:** | Sub type objects may be used wherever objects of a super type are expected. This is known as substitution. |
| **Sub typing:** | As an analog to the class hierarchy, types form a type hierarchy. The type defined by a subclass definition is a sub type of the type of it's super class. |
| **Polymorphic Variables:** | A variable that can refer to a variety of objects. |
| **Extends:** | Used to assign a parent to a class. |
| **Super:** | Used to refer to the parent of this objects object. |
| **Cast:** | Casting lets the compiler know what object will go there at run time. |
| **Object:** | All classes with no explicit superclass have Object as their super |

|  |  |
| --- | --- |
| Chapter 11 | |
| **Static Type:** | The static type of a variable v is the type as declared in the source code in the variable declaration statement. |
| **Dynamic Type:** | The dynamic type of a variable v is the type of the object that is currently stored in v. |
| **Overriding:** | A subclass can override a method implementation. To do this, the subclass declares a method with the same signature as the super class, but with a different method body. The overriding method takes precedence for method calls on subclass objects. |
| **Interface:** | A Java interface is a specification of a type (in the form o fa type name and a set of methods). It often does not provide an implementation for most of its methods. |
| **Dynamic method lookup:** | Checks each object in the hierarchy starting with the calling object. |

|  |  |
| --- | --- |
| Chapter 12 | |
| **Abstract classes:** | An abstract class is a class that is not intended for creating instances. Its purpose is to serve as a super class for other classes. Abstract classes may contain abstract methods. |
| **Multiple inheritance:** | A situation in which a class inherits from more than one super class is called a multiple inheritance. |
| **Abstract:** | Not to be called directly, used for inheritance. |
| **Implements:** | Notation that allows an interface to become the parent of an object. |
| **Interface:** | Describes what a class is and what it can do without showing implementation. Generally, includes several items which need to be overwritten. |