**Java Training Index**

**05-08-2024**

1. Languages & Packages
2. Java Features
   1. Why Java is platform independent
3. JDK, JRE, JVM
4. Basic Java
   1. Datatypes
   2. Operators
   3. Conditions(if, if-else)
   4. Loops(Nested for loops)
5. Packages

CoreJAVA-Workspace :

CoreJAVA-Development

CoreJAVA-Application

**06-08-2024**

1. Logical Programming
2. Patterns
3. Switch Case
4. Arrays
   1. 1D Array's
   2. 2D Array's
5. Enum's
6. Event Management System

**07-08-2024**

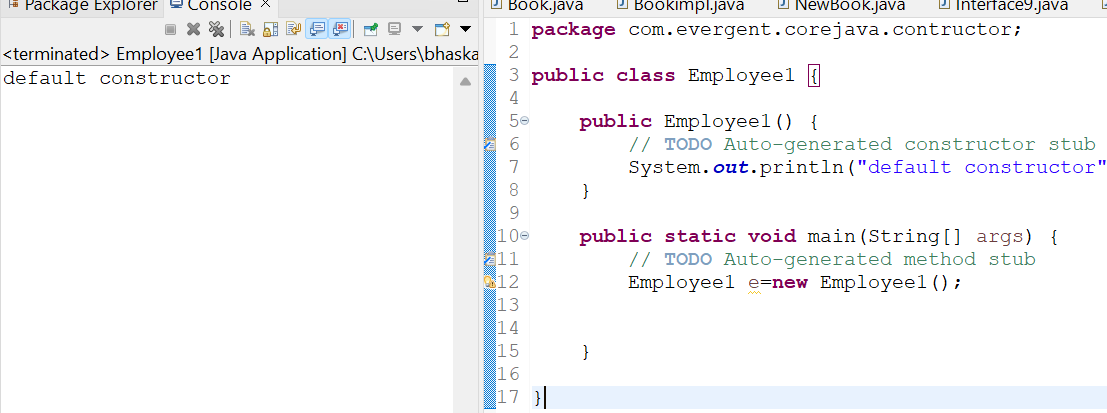
1. OOPS
   1. Encapsulation
   2. Inheritance
   3. Polymorphism
   4. Abstraction
   5. Method Flows
   6. IS-A and HAS-A relationship
2. System class

**08-08-2024**

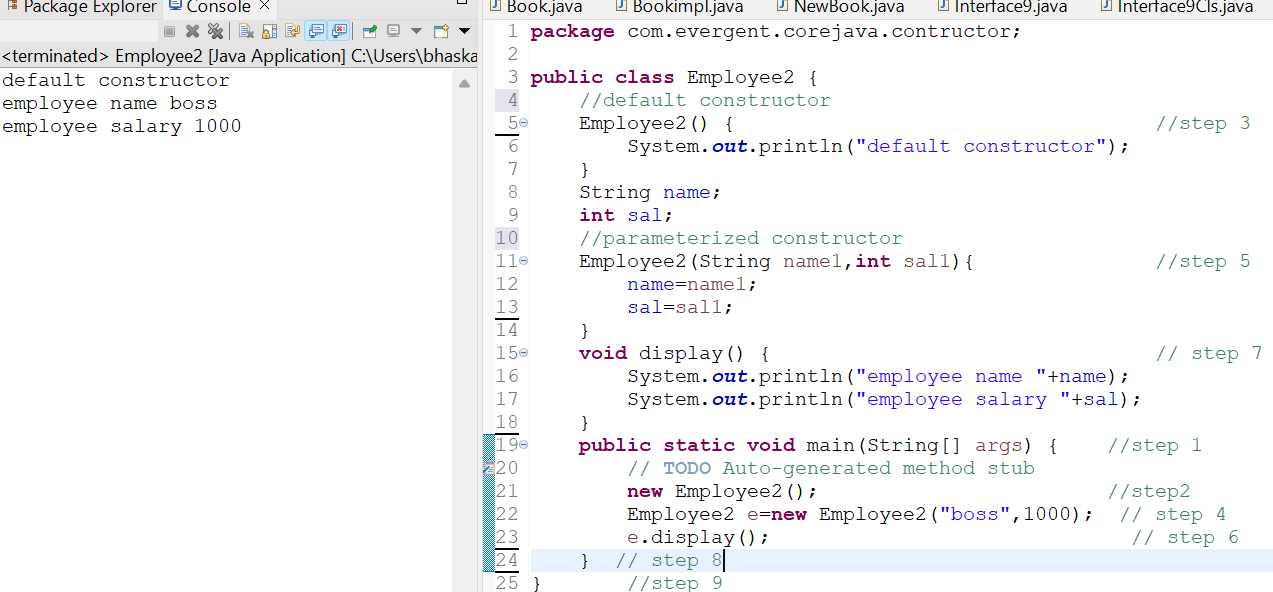
**Constructors**

* Constructors are used to initialize the objects
* Having same name that as class Name
* 2 types of constructors

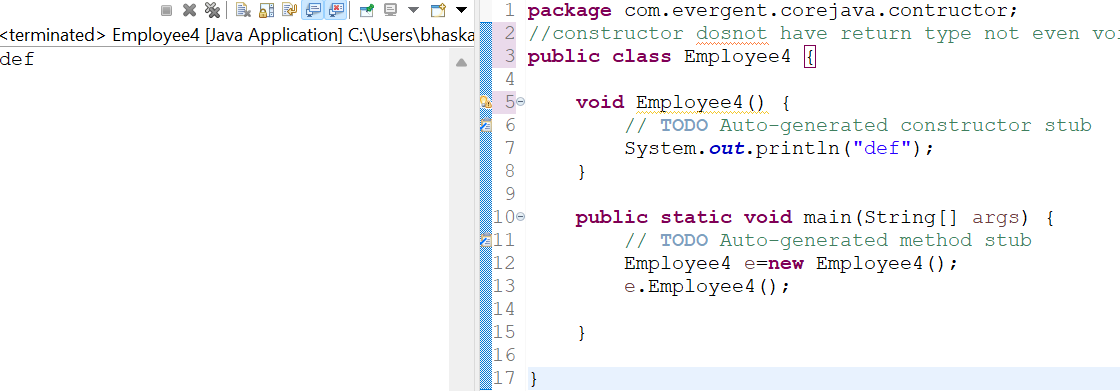
1. Default constructors - No parameters, compiler defined the default constructor implicitly during compilation



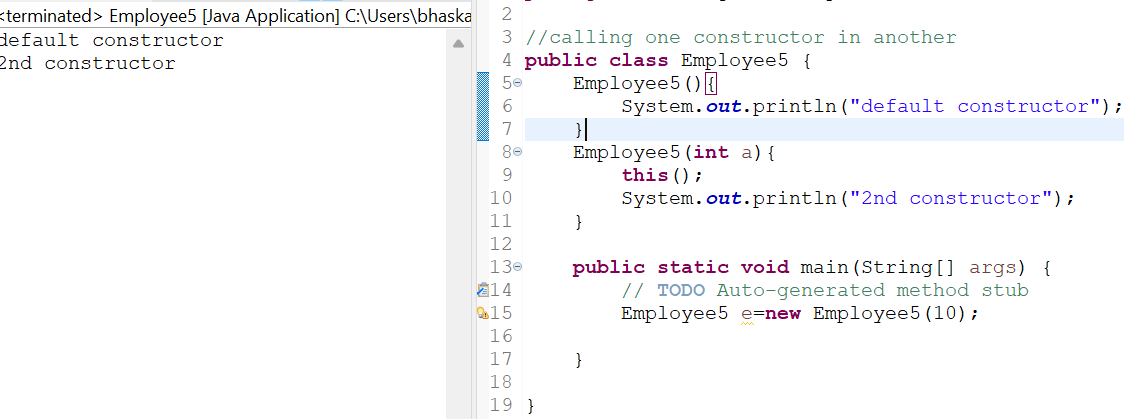
1. Parameterized constructor - Constructor consist of parameters



* Constructor doesn’t have return type not even void,else it act as a method



* Calling one constructor in another constructor by ‘this()’ keyword

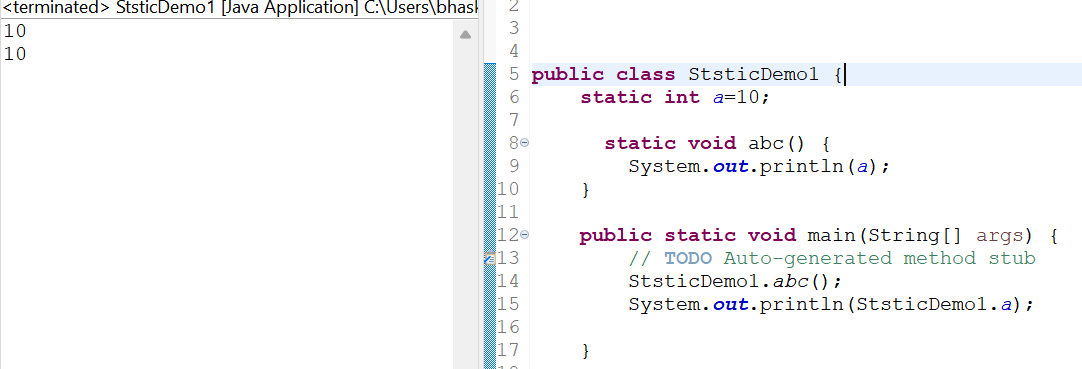


**09-08-2024**

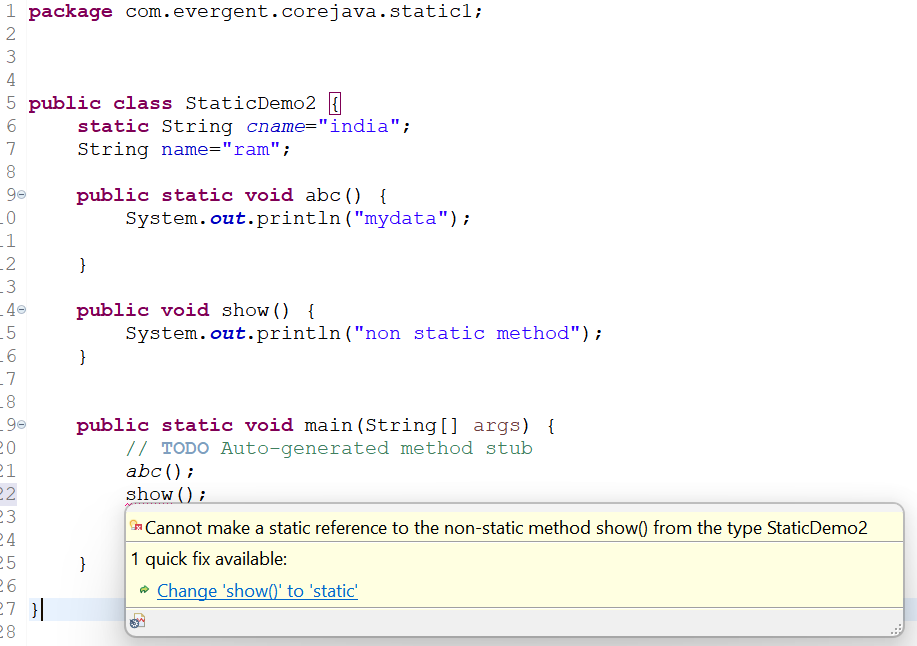
1. Static and Final

**Static**

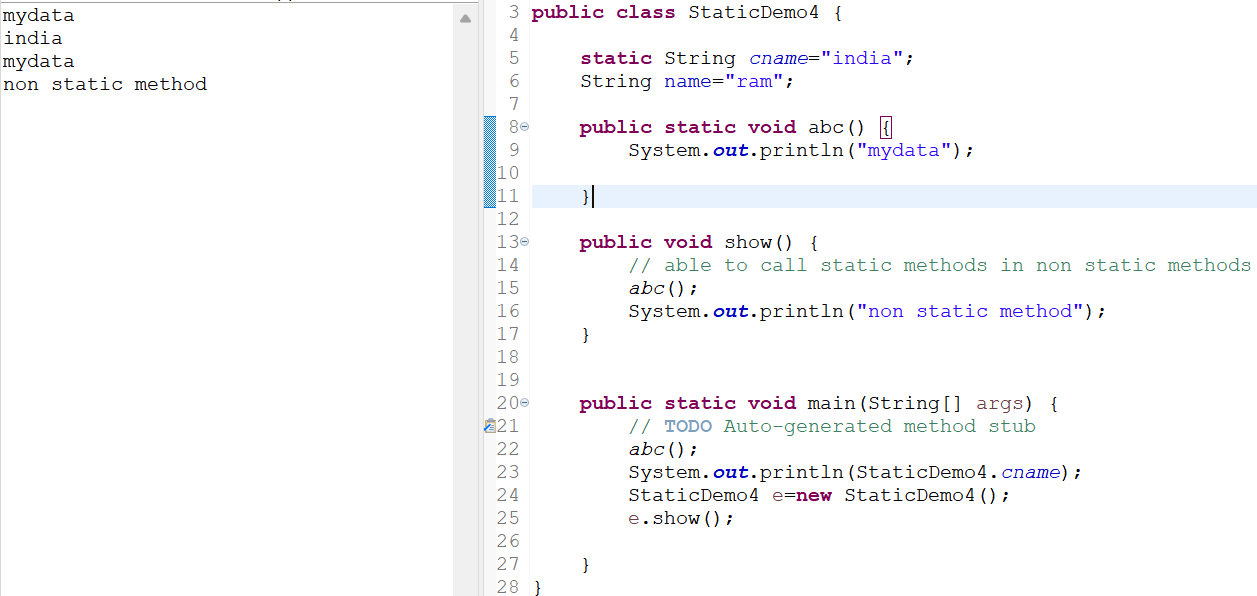
* Static is a Keywod
* We can declare static with variables and methods
* We can call the static var or methods by their class name
* Static methods can able to access only static members
* Non static methods can access static var and static methods
* Static variables are belong to class not object



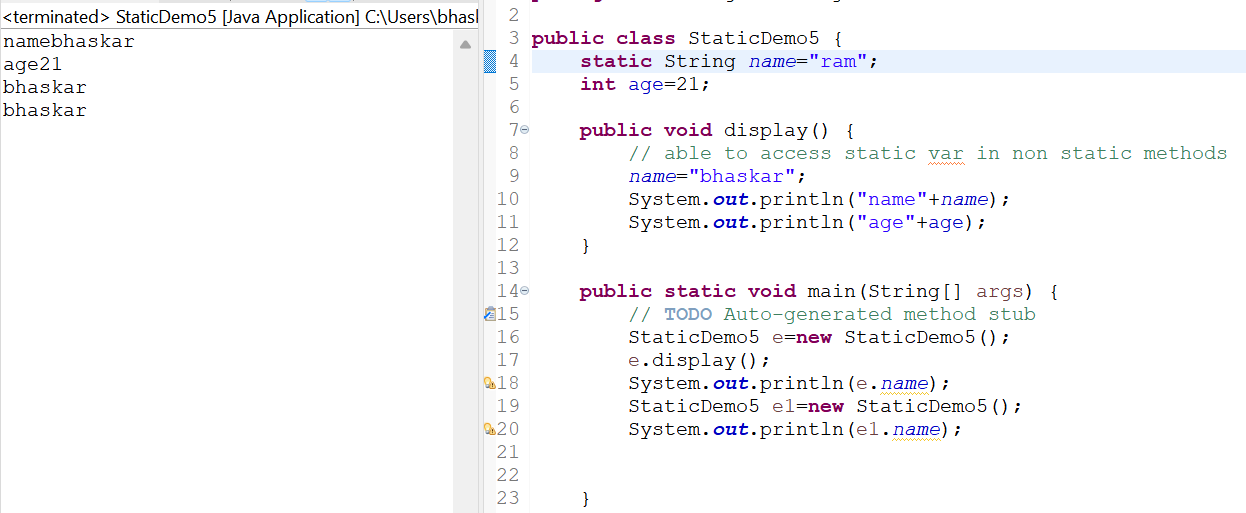
Static methods cannot call non-static methods



Non static methods ca call static methods

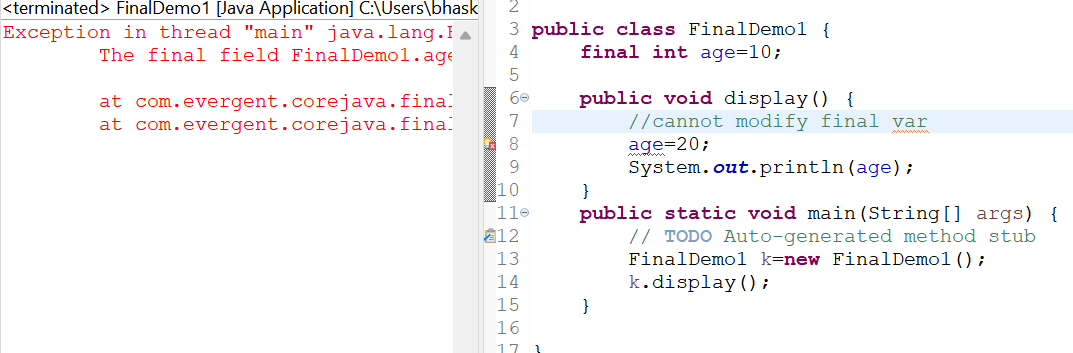


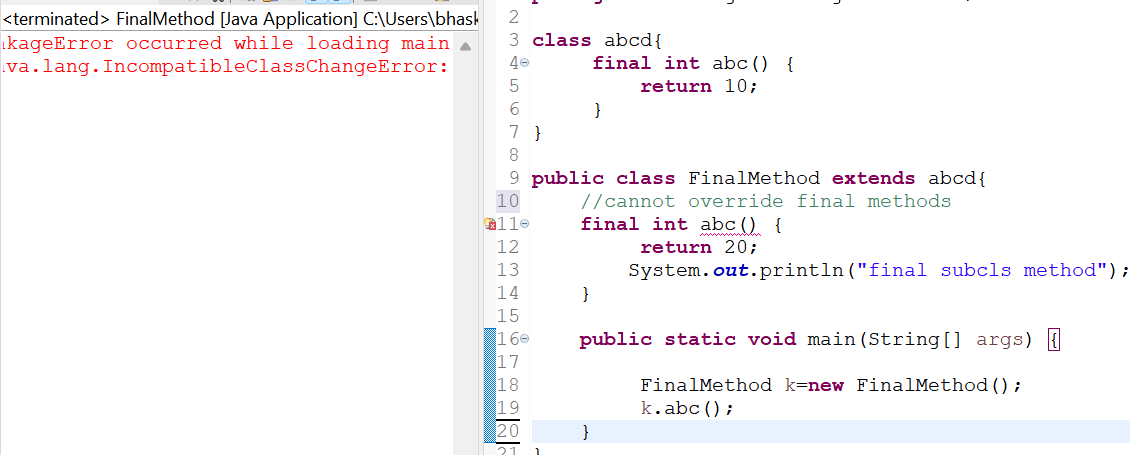
Able to access static var in non static methods



**Final**

* Final isa Keyword
* We can declare final keyword with variables,methods and classes
* Final variables cannot modified

****

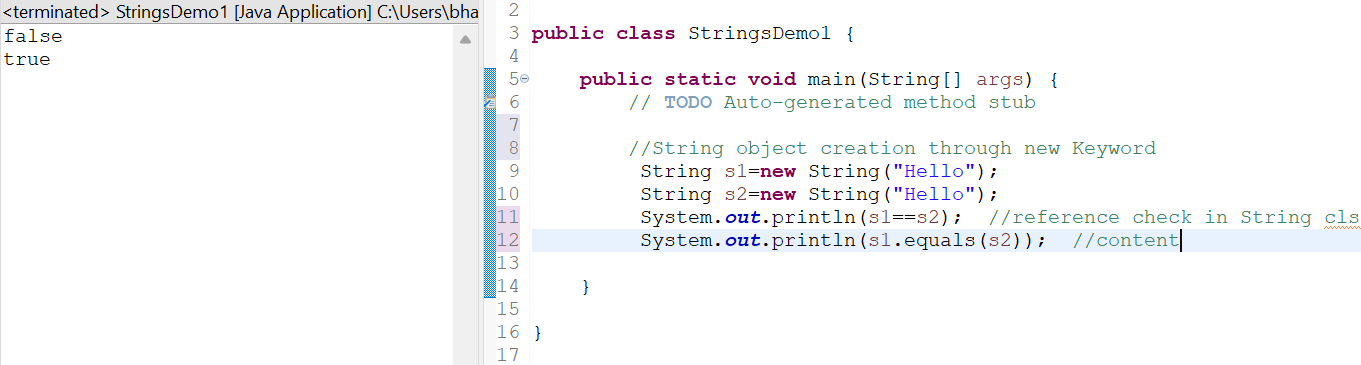
* final methods cannot override
* ****

**12-08-2024**

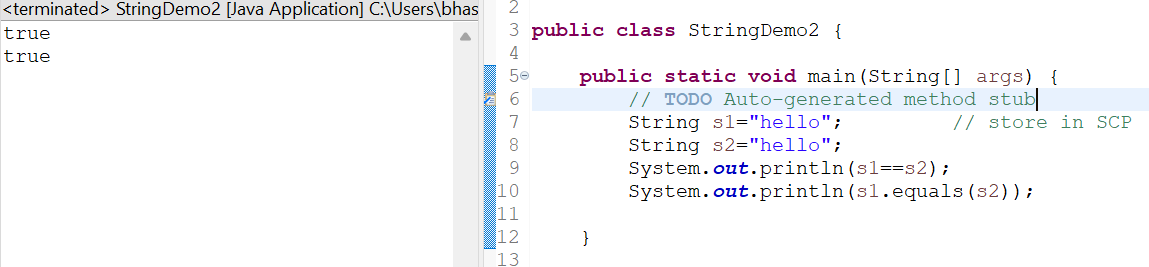
**String**

* String is a sequence of characters which represent a text
* Strings are immutable, once we create a String obj we cannot modify it, if we modify it a new object is created and previous object is garbage collected
* Strings are non synchronized (not thread safe)
* String class if final
* We can able to create Strings in 2 ways

1.By **new** keyword (by this 2 objects are created on in heap area and one in String constant pool)



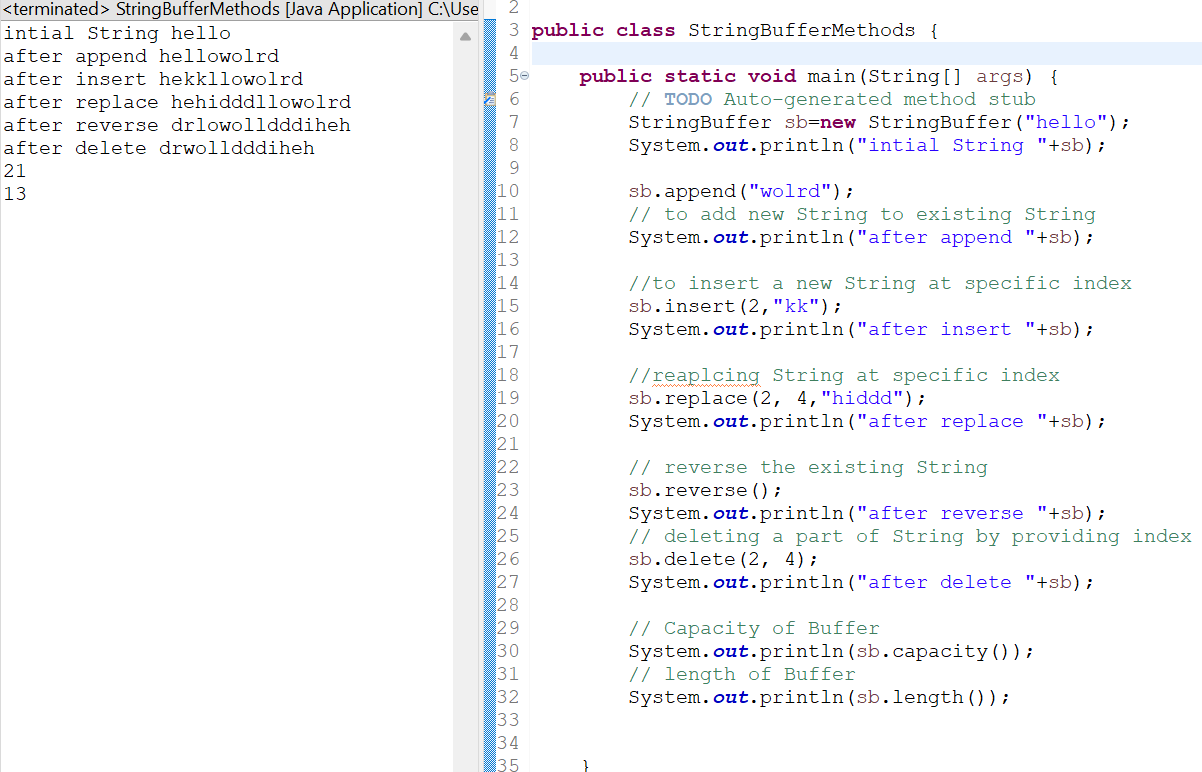
1. By String **Literal**



**StringBuffer**

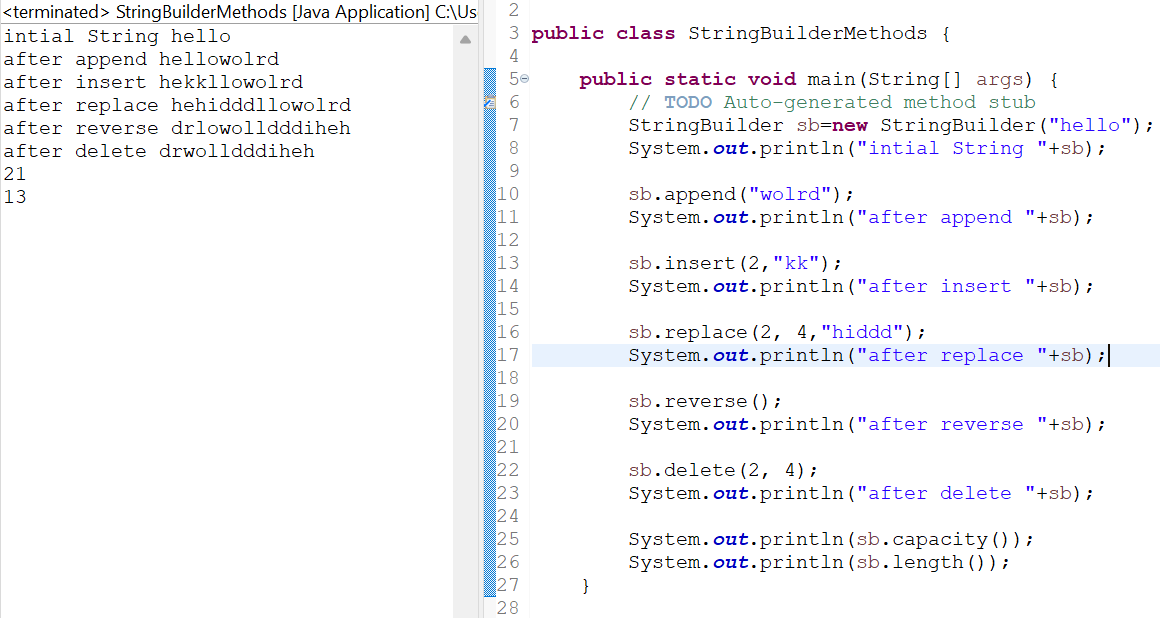
* String Buffer is a final class
* Mutable
* Synchronized (thread safe)
* Consist of methods like

append() reverse(); replace(); ,insert(),length(),capacity(),delete()



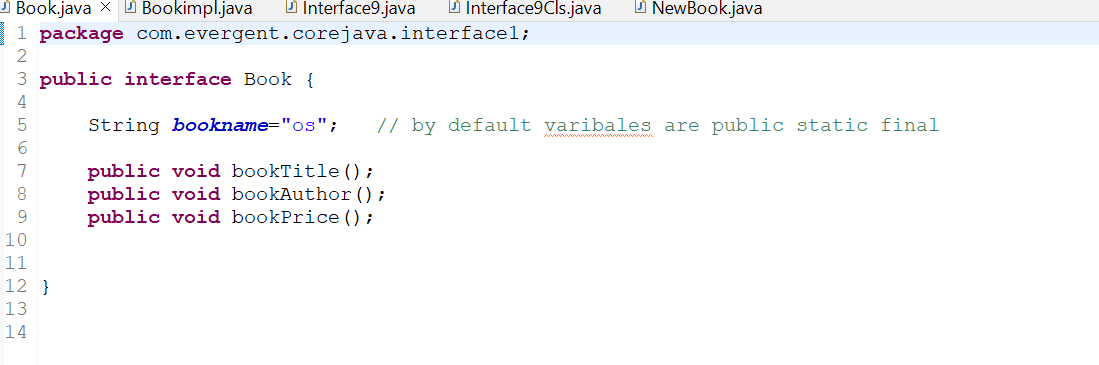
**String Builder**

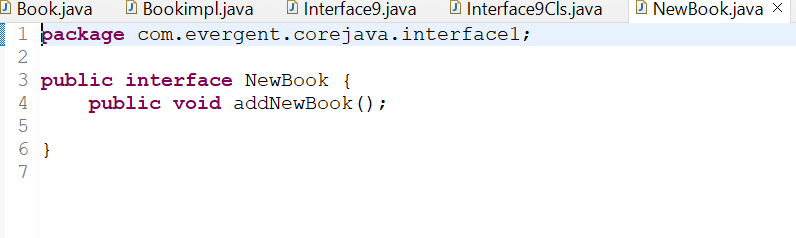
* String Builder is final class
* Mutable
* non synchronized(not thread safe)

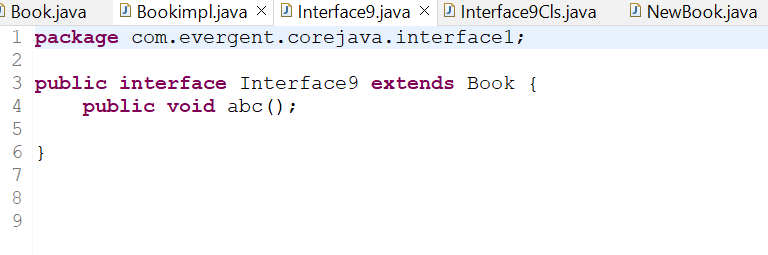
****

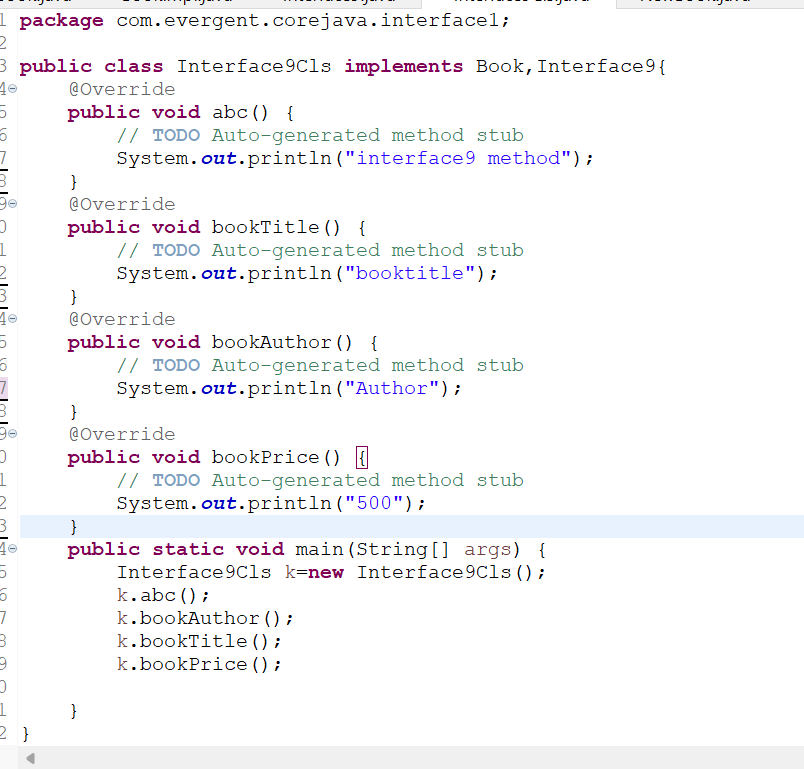
**Day 7: 13-Aug-2024:**

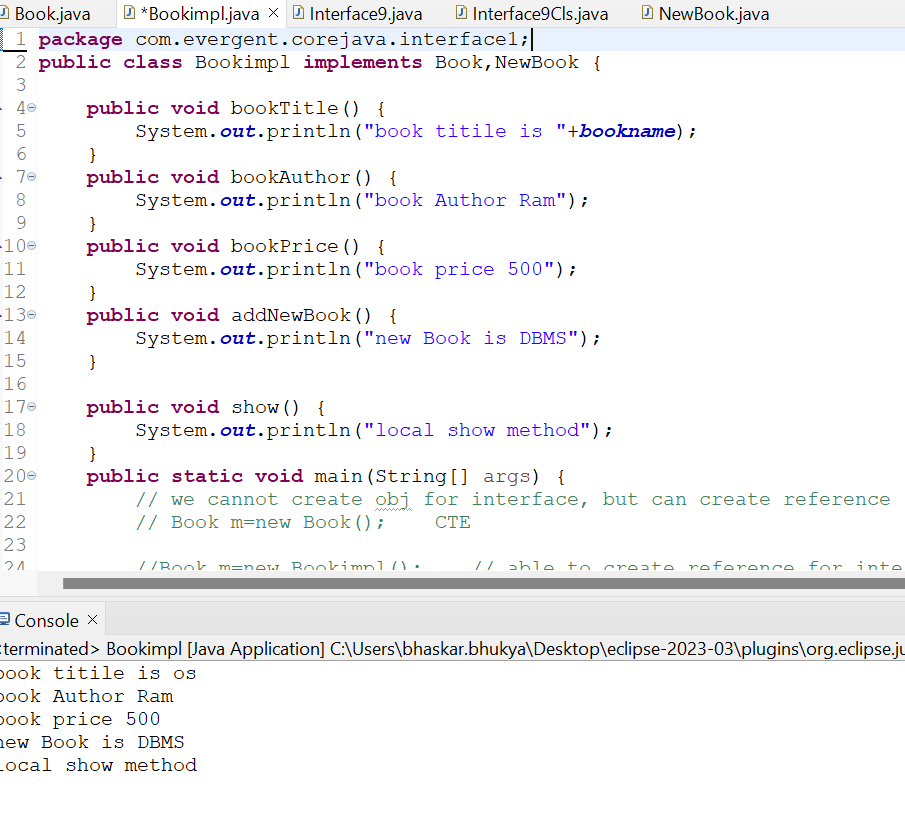
1. Interfaces:
   1. Interface is a keyword.
   2. We can declare method signature only but not implementation.
   3. By default all interface methods are abstract.
   4. If any class implements the interface then that class should be override all interface method, otherwise that class will be showing compile time error.
   5. We cannot create object to interface but we can create reference to interface.
   6. We can declare variables inside interface - by default (public static final) variables.
   7. JAVA will support multiple inheritance through interface.
   8. One class can implement interfaces.
   9. One interface extends other interfaces.
   10. We can declare interfaces without methods is called Marker Interfaces.
   11. Example:
       1. clonable
       2. serializable





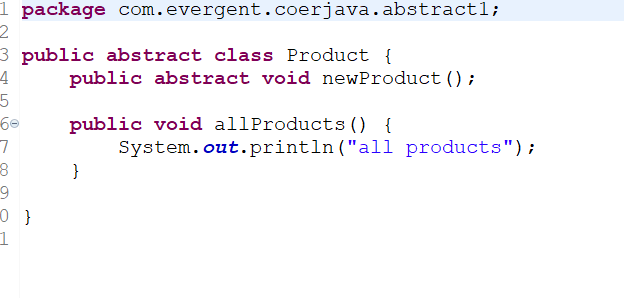


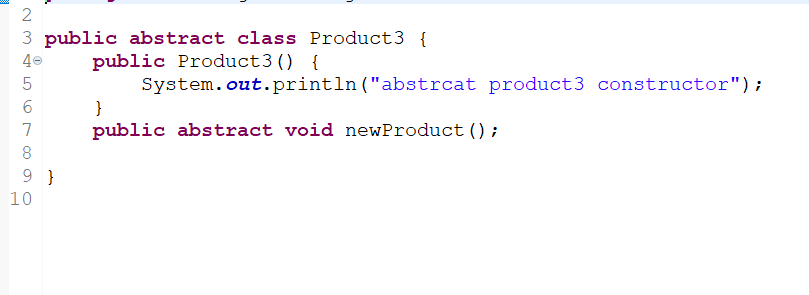


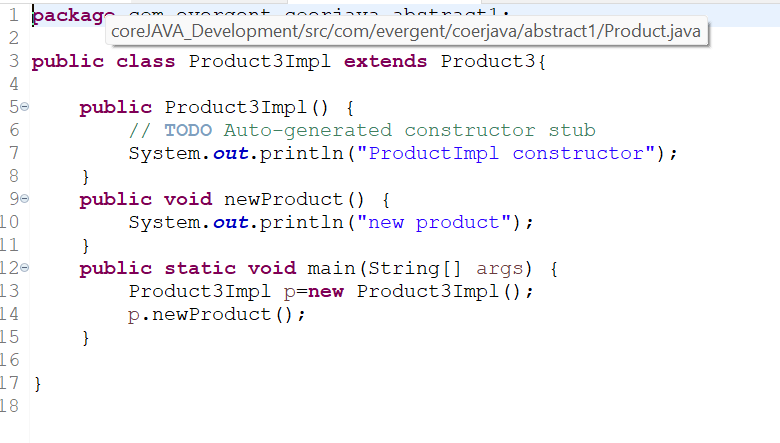


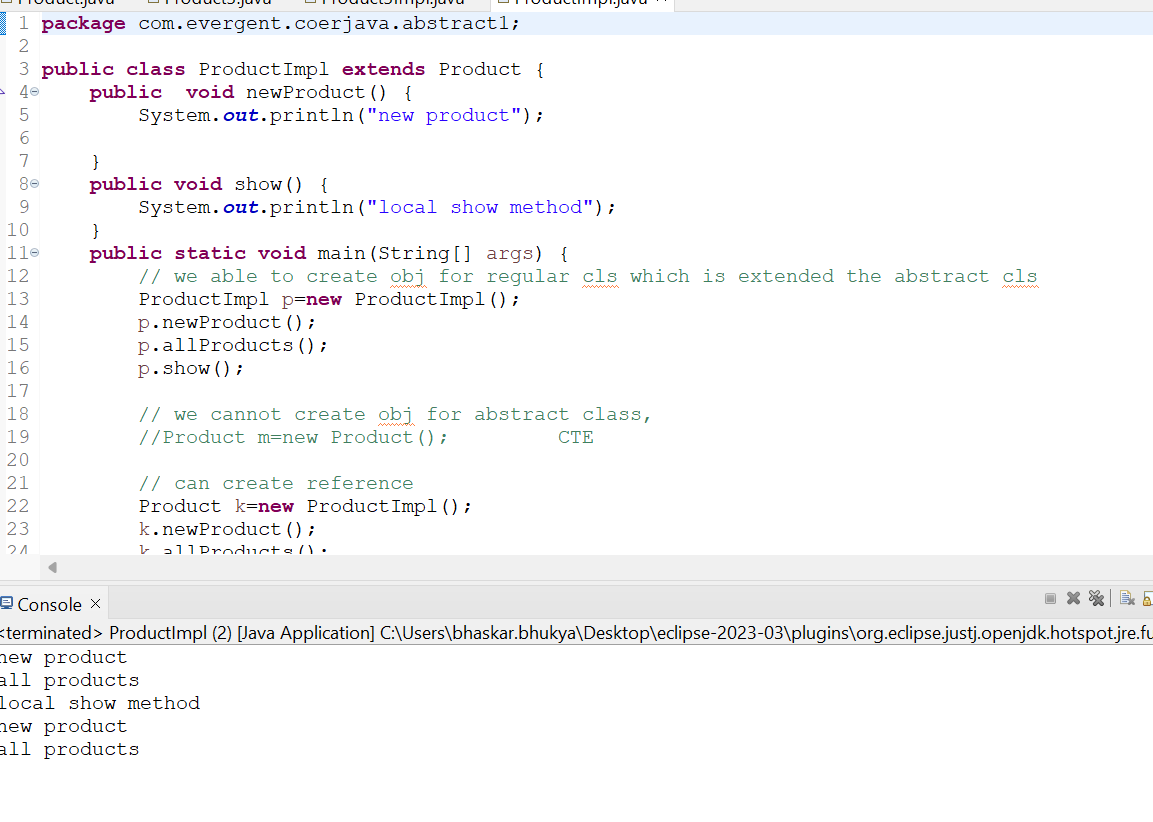
**Day 8: 14-Aug-2024:**

1. **Abstract class:**
   1. Abstract is a keyword.
   2. Abstract class can have abstract methods and concrete methods.
   3. If any class having one abstract method that class should be declared as a abstract keyword, otherwise that class will be showing compile time error.
   4. If any class extends abstract class that class should override all abstract methods, otherwise the class will be showing compile time error.
   5. We cannot create object to abstract class but we can create reference to abstract class.
   6. We have to declare abstract methods explicitly.
   7. We can create constructor to abstract class.
   8. We can access abstract class constructor through the sub-class object creation.



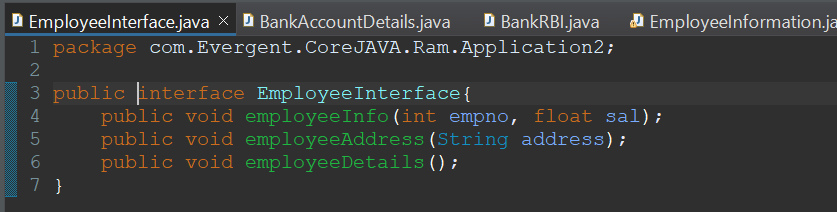


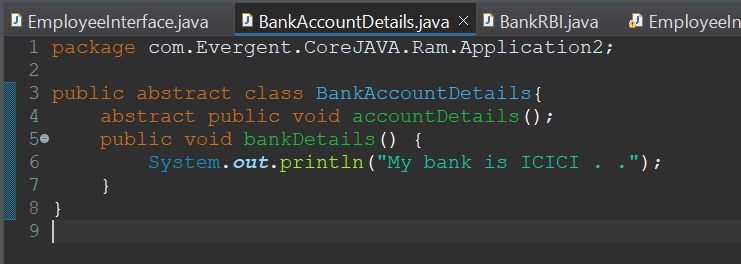


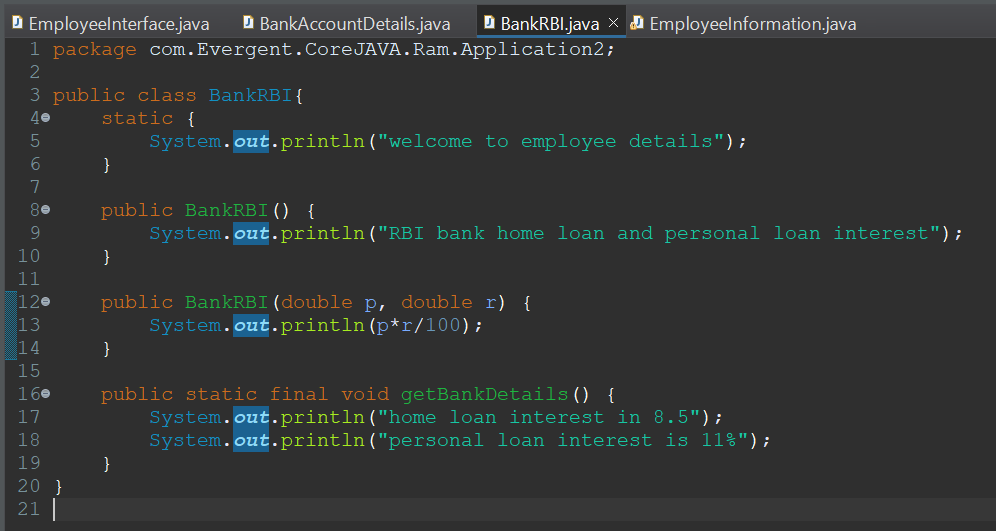


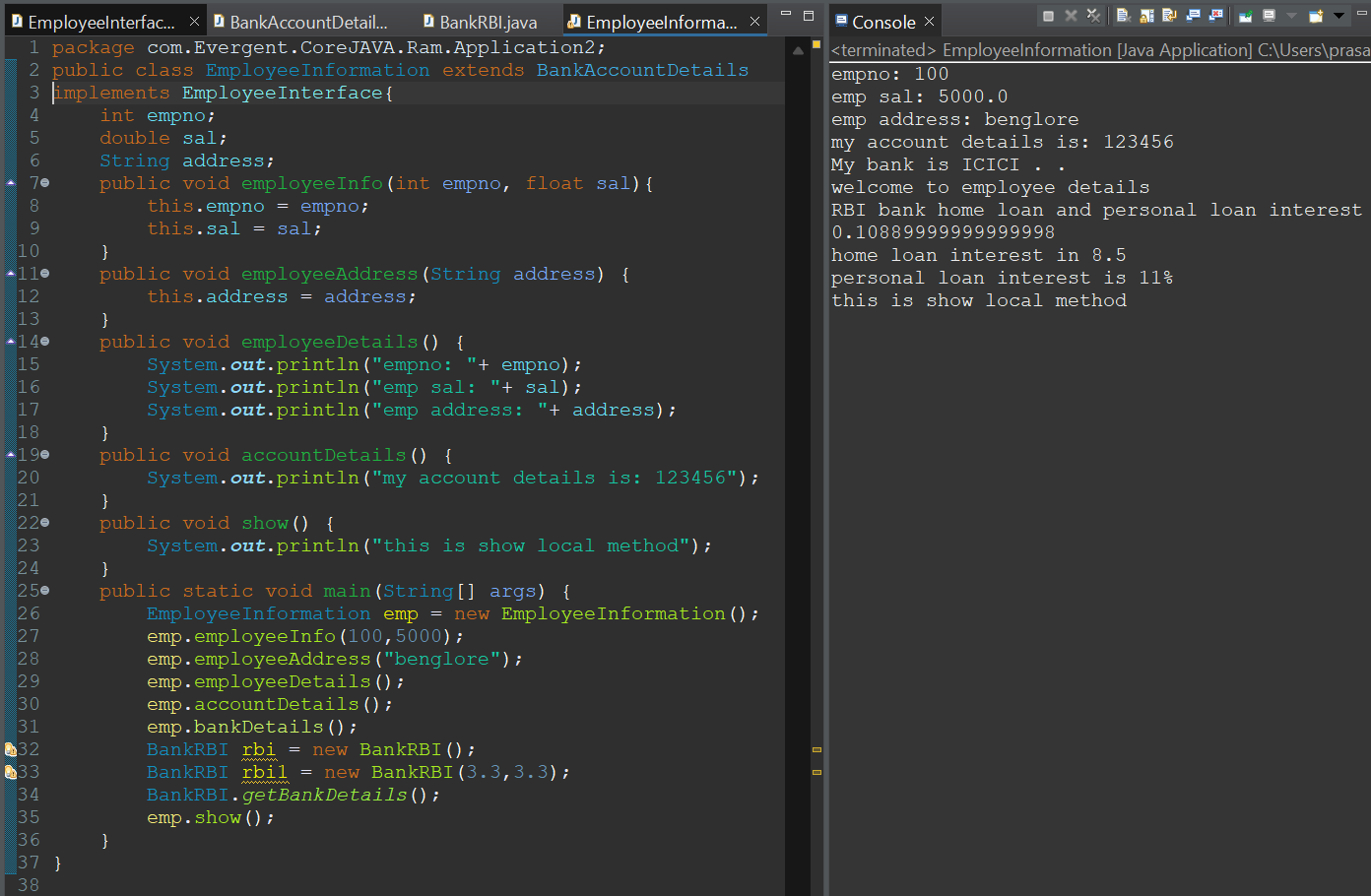
**Day 9: 16-Aug-2024:**

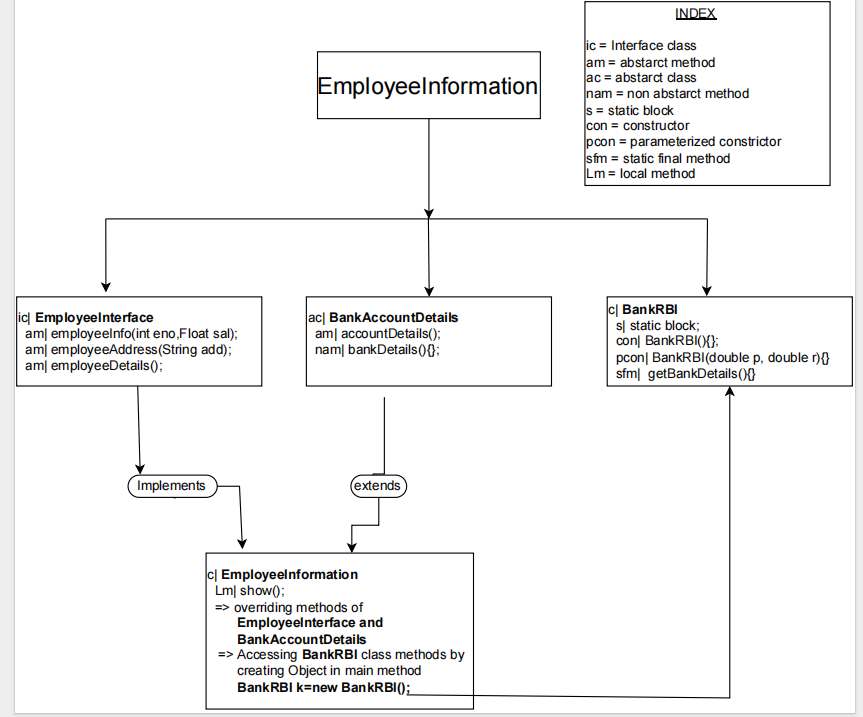
1. Java Concept Test
2. Application design and code







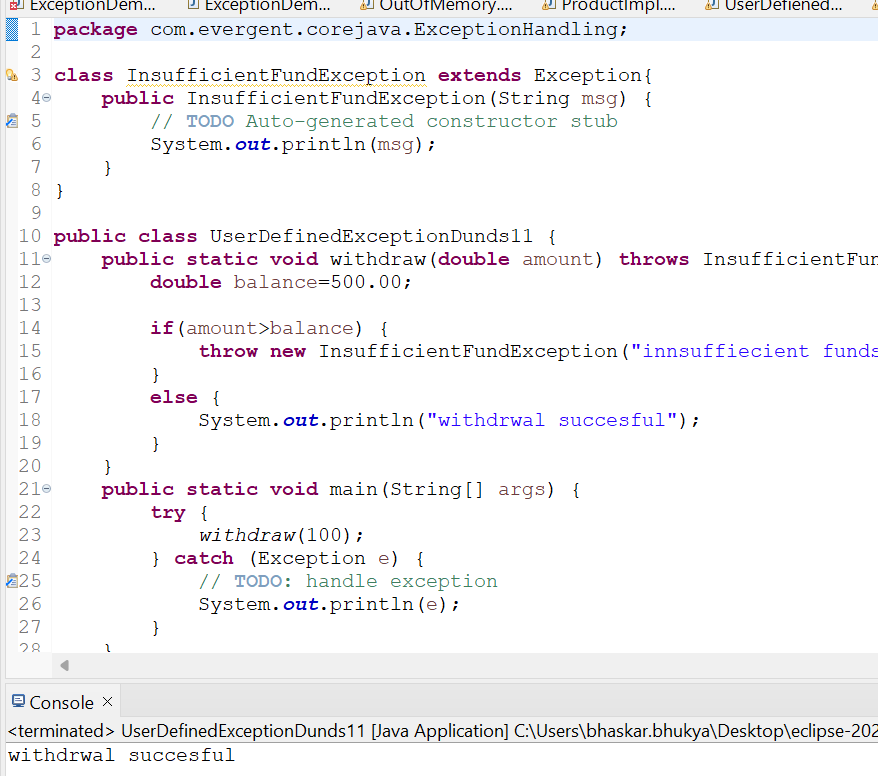




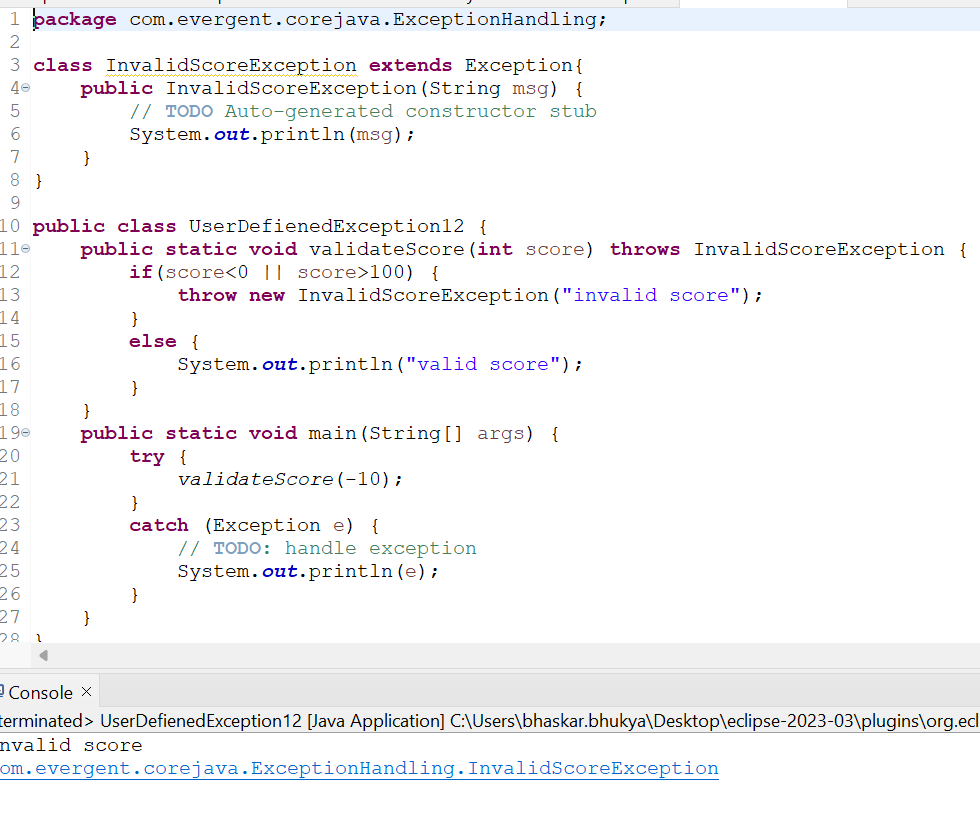
**Day 10: 19-Aug-2024 and Day 11: 20-Aug-2024:**

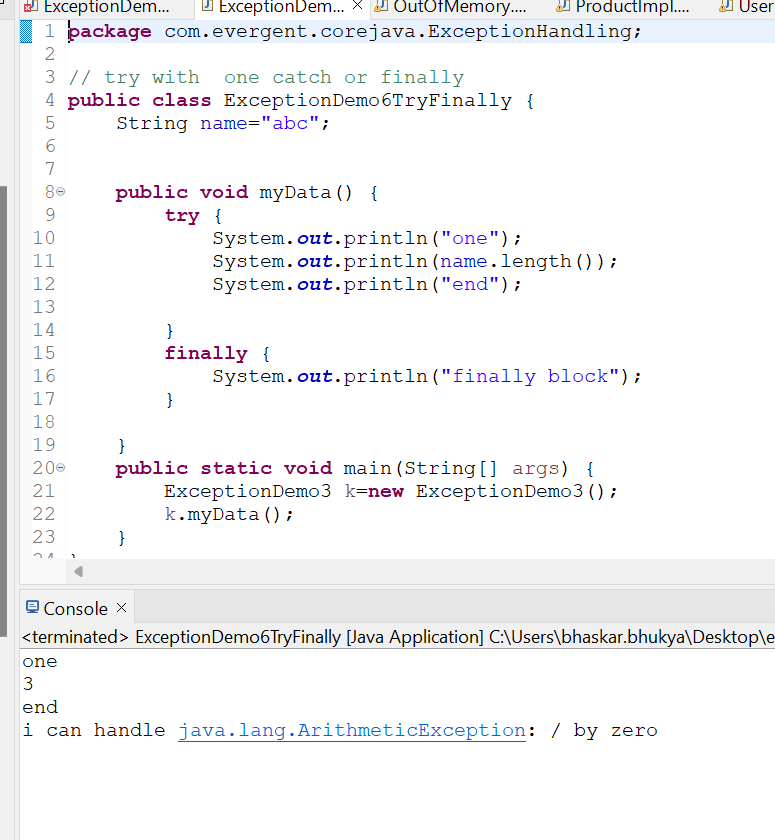
1. **Exception Handling:**
   1. Exception handling is mechanism
   2. Exceptions are inbuilt mechanism of JAVA
   3. All exceptions are executed while abnormal conditions
   4. Normal flow it won’t execute any exception
   5. Once any exceptions are occurring in java then remaining lines of code are unreachable.
   6. Java.lang.Throwable is super class for exception and error
   7. There are two types of exceptions in JAVA:
      1. Checked exception
      2. Unchecked exception
   8. All checked exceptions are compile time exceptions
   9. All unchecked exceptions are run time exceptions
   10. There are 5 keywords in exception handling:
       1. try
       2. catch()
       3. finally()
       4. throws
       5. throw
   11. Try is for business logic
   12. Catch is for handling exceptions
   13. Finally block, is executed if exception occurs or not
   14. Throws an exception will be executed method by method
   15. Throw is for run time exceptions & will call predefined exceptions
   16. Try followed by either catch or finally block
   17. We should follow exception hierarchical
   18. We can create our own (user-defined) exceptions
   19. Our own exceptions extends exception or run time exception
   20. All exception classes are in to java.lang package
   21. There is two exceptions in class, developer should be handle 1st exception then after 2nd exception will be handled.
   22. Errors cannot be controlled

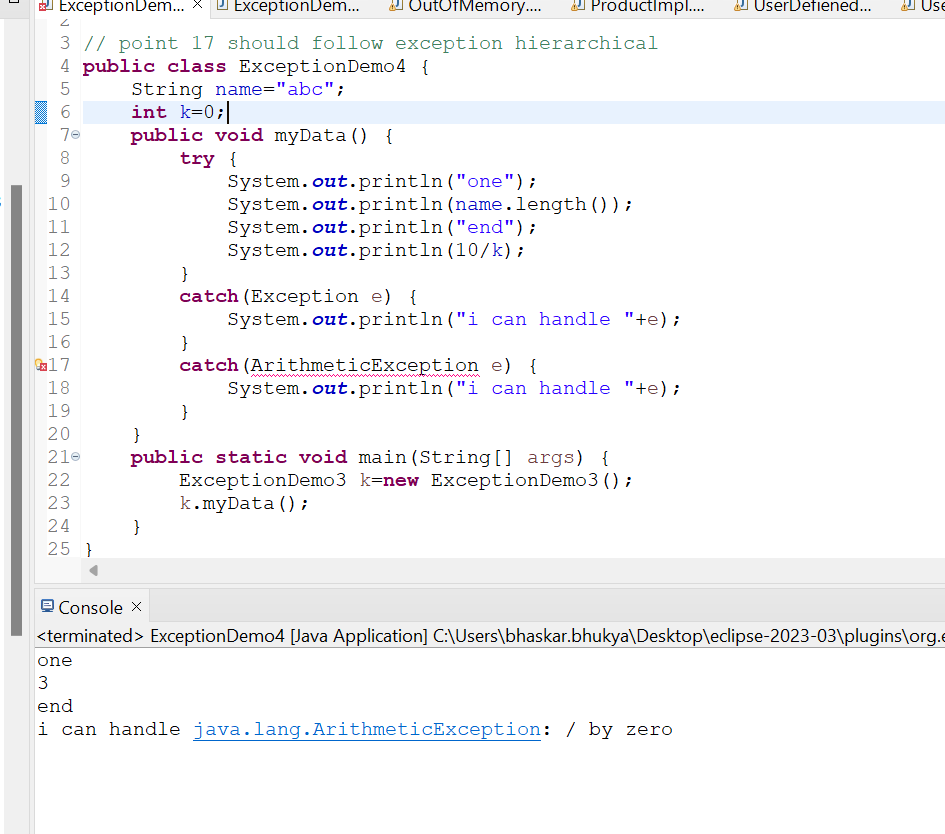












**JAVA Beans:**

1. Java bean is a mechanism
2. Java bean is light weight
3. All attributes are private
4. get/set methods are public
5. Implements java.io.Serializable interface
6. We can achieve tightly encapsulation through java beans

