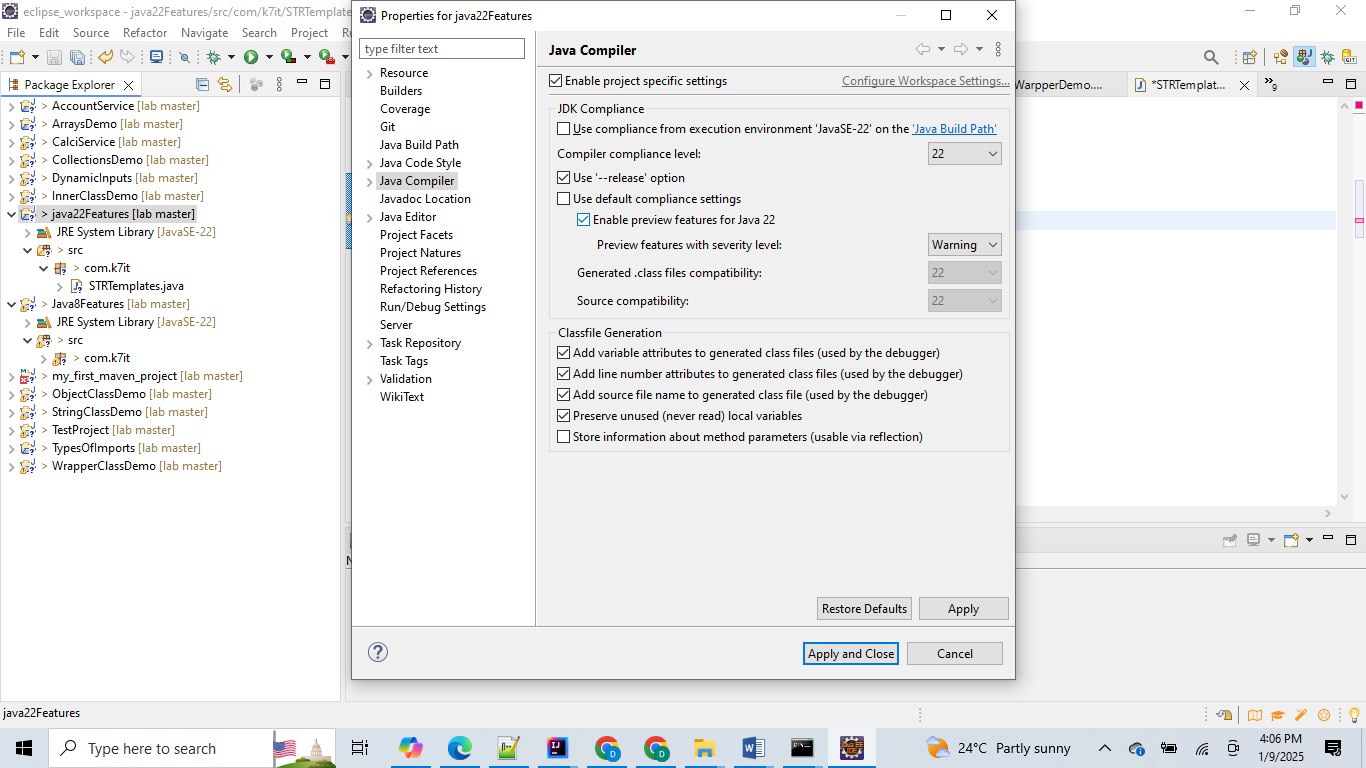
Java 22 features:  
================  
  
1.enhancement on instanceof key words and switch cases   
  
  
old :  
=======  
  
   package com.k7it;   
  
record Account(int i,int j) {}   
  
public class InstanceOfDemo21 {  
 public static void main(String[] args) {  
 Object obj  = new Account(10, 20);  
 System.out.println( add(obj));  
   
   
 }  
   
   
static int add(Object obj) {  
   
 if(obj instanceof Account a) {  
 return a.i() + a.j();  
 }  
 return 0;  
}   
  
}   
  
new :   
  
  package com.k7it;   
  
record Account(int i,int j) {}   
  
public class InstanceOfDemo21 {  
 public static void main(String[] args) {  
 Object obj  = new Account(10, 20);  
 System.out.println( add(obj));  
   
   
 }  
   
   
static int add(Object obj) {  
   
 if(obj instanceof Account(int i, int j)) {  
 return i+j;  
 }  
 return 0;  
}   
  
}

Exaple2 :

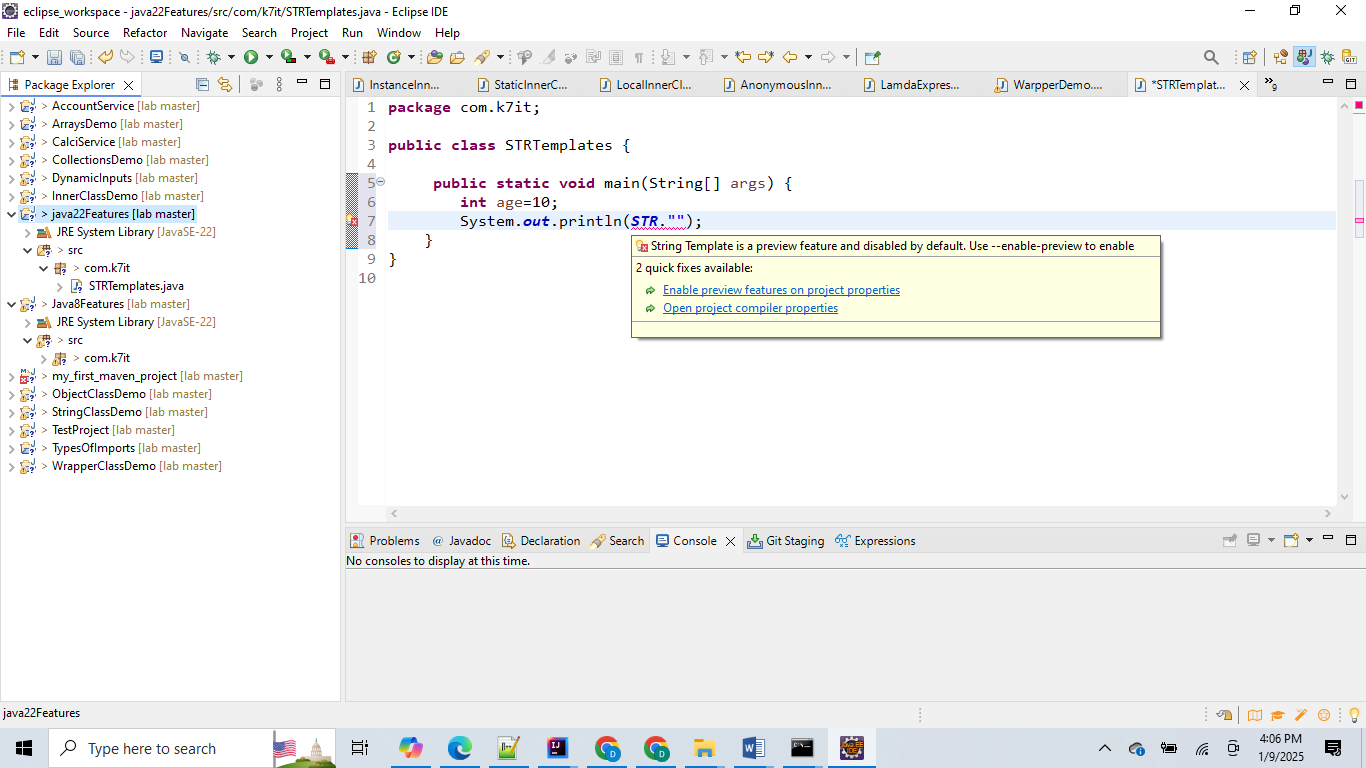
package org.example.com.k7it;  
  
public record Person(int age,String name) {  
}

package org.example.com.k7it;  
  
import java.util.function.Predicate;  
  
public class PersonManager {  
 public static void main(String[] args) {  
 Object obj = new Person(10,"java");  
 if(obj instanceof Person p){  
 System.*out*.println(p.name()+","+p.age());  
 }  
  
 if(obj instanceof Person(int age, String name)){  
  
 // System.out.println(name+","+age);  
 System.*out*.println(*STR*."\{name},\{age}");  
 }  
 }  
}

       
**2. enhancement on STR templates :** to use this one we need to upgrade our eclipse with latest version 2024-06 or above versions and java 22.  
    String Template is a preview feature and disabled by default. Use --enable-preview to enable  
 for this in eclipse we can select quick fix options help to enable preview option on project properties .



Or



STR template internally it will use  the process() and it will return final string as output from process().  
  
    package com.k7it;   
  
public class TestDemo {   
  
 //@SuppressWarnings("preview")  
 public static void main(String[] args) {  
 String name = "K7iT";  
 int age = 25;  
 double ht = 5.5;  
 double wt = 68.25;  
 System.out.println(STR."My name is \{name} , my age is \{age}, my wt is \{wt}, my ht is \{ht}");  
   
 }  
}   
  
3. unnamed variables and attributes  :  
===============================  
where ever we are using any variables with temporary scope or single line scope or the variables which are not assigning or not passing as reference or parameters those type of variables we can declared as unnamed variables.   
  
   example:  
      
 try{  
    int i = 10/0;  
 }catch(ArithmeticException e){  
    Sop("somthing divide by zero not allowed");  
 }  
   
 try(FileReadr fr = new FileReadr("path");  
     BufferedReader br  = new BufferedReader(fr);  
    ){  
   
 }  
   
 switch(a){  
    case SavingAccount s -> sop("accout type is saving");  
    case CurrentAccount c -> sop("accout type is Current");  
 };  
   
 same code we can rewrite with unName variables:  
 ==============================================  
         
    try{  
    int i = 10/0;  
 }catch(ArithmeticException \_){  
    Sop("somthing divide by zero not allowed");  
 }  
   
 try(FileReadr fr = new FileReadr("path");  
     BufferedReader \_  = new BufferedReader(fr);  
    ){  
   
 }  
   
 switch(a){  
    case SavingAccount \_ -> sop("accout type is saving");  
    case CurrentAccount \_ -> sop("accout type is Current");  
 };  
   
  
4. **implicitly declared classes and instance main method:**

define the class with out class  key word and run our code with direct simplifier main() like   
     void main(){} instead of using public static void main(String args[]).     
 to run this also we need to enable preview option and source  like bellow  
   
 E:\src>java --source 22 --enable-preview HelloWorld.java  
   
 example :

HelloWorld.java   
   
 ===================  
    void main(){  
           System.out.println("Hello world");  
        }  
   E:\src>java --source 22 --enable-preview HelloWorld.java  
     
   output: HelloWorld.  
     
  Note:  --source 22 --enable-preview  this option is mandatory for implicitly declared classes and instance main().  
   
**5. Running multiple source files  directly without using javac cmd**.   
     : java --source 22 --enable-preview .java file name and cmd line args.  
   
   
B.java  
public class B {  
  public static void main(String args[]){  
    A a1 = new A();  
 a1.p  
 rint("Sai kumar");  
 C c1 = new C();  
 c1.print("Tharun kumar");  
  }  
}   
A.java  
public class A{  
  void print(String s){  
    System.out.println(s);  
   }  
}  
C.java  
public class C{  
  void print(String s){  
    System.out.println(s);  
   }  
}   
  
 src> java --source 22 --enable-preview B.java   
 output:  Sai kumar  - > A class output   
           Tharun Kumar - > c class output  
     
   here all 3 .java files will compile automaticaly and run internally.   
     
**9. statements before super() function inside constructor body:**

from java 22 onwards we can keep super() any where in the constructor body explicitly based on user requirements.   
   
   package com.k7it.superstmt;   
  
 public class Account {  
 int balance;   
  
 Account(int balance) {  
 System.out.println("I am at account class constructor");  
 this.balance = balance;  
 }  
   
   
 }  
  
  
 class SavingAccount extends Account {  
 int accNo;  
 int balance;   
  
 SavingAccount(int accNo, int balance) {  
 System.out.println("I am a saving accout constructor");  
 super(balance);// super statement at line 2  
 this.accNo = accNo;  
 }  
   
   
 }  
  
        
   public class AccountManager {  
 void main(String args[]) {  
 System.out.println(args[0]);  
 SavingAccount sb = new SavingAccount(123, 10000);  
 }  
}   
  
output :   
  
instance main() and super statemnt any where in the constructor body  
I am a saving accout constructor  
I am at account class constructor  
  
  
here : void main(String args[]) : string args[] required only if we need to read any cmd line argments else just   
       void main(){}  to execute main().  
   
   
**6. sequenced collection and sequenced set and sequenced map interfaces in collection frameworks:**  
**7. enhacements in Stream api : geather()method ,  Geathers class**   
**8. enhacements in Vector class  
10. scopped values in multiple threads as thread local**