Palindrome Program :

**import** java.io.\*;

**public** **class** Palindrome {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

String s1="NITIN";

String s2="";

**for**(**int** i=s1.length()-1;i>=0;i--)

{

s2=s2+s1.charAt(i);

}

**if**(s1.equals(s2)) {

System.***out***.println("Palindrome");

}

**else**

{

System.***out***.println("Not a Palindrome");

}

}

}

**char** a[]=s1.toCharArray(); --Converts String to Character Array

Anagram :

**import** java.util.Arrays;

**public** **class** Anagram {

**public** **static** **void** main(String[] args) {

String s1="race";

String s2="Care";

s1=s1.toLowerCase();

s2=s2.toLowerCase();

**char** a[]=s1.toCharArray();//Convert the String s1 to Char Array

**char** b[]=s2.toCharArray();

Arrays.*sort*(a);

Arrays.*sort*(b);

**boolean** i=Arrays.*equals*(a, b);

**if**(i)

{

System.***out***.println("Anagram");

}

**else**

{

System.***out***.println("Not a Anagram");

}

}

}

Panagram :

Such a String that contains all the Alphabets.

**public** **class** Panagram {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

String s1="The quick brown fox jumps over the lazy dog";

s1=s1.replace(" ","");

s1=s1.toLowerCase();

**boolean** val=**true**;

**int** a[]=**new** **int**[26];

**for**(**int** i=0;i<s1.length();i++)

{

a[s1.charAt(i)-97]++;

}

**for**(**int** i:a)

{

**if**(i==0)

{

val=**false**;

**break**;

}

}

**if**(val==**true**)

{

System.***out***.println("Panagram");

}

**else**

{

System.***out***.println("Not a Panagram");

}

}

}

OOPS :

Encapsulation ----private/Security---Data hiding/Data binding

Inheritance --- Code Reusability

Polymorphism --- Code Flexibility

Abstraction --- implementation hiding only feature visibility

Encapsulation:

Data hiding

Data Binding

Providing Security

Provide controlled access to members.

**class** Student

{

**int** age;

String name; ///Data Members//

String city; //We should not provide direct access to data members.

//there might who is using object might give incorrect values.

}

**public** **class** Encap {

**public** **static** **void** main(String[] args) {

Student s=**new** Student();

s.age=20; //We are having direct access to data members and we can update the values of it.

s.name="Chinmay";

s.city="Banglore";

}

}

Method takes the value from different class and set values to it’s data member then we call method as setter .

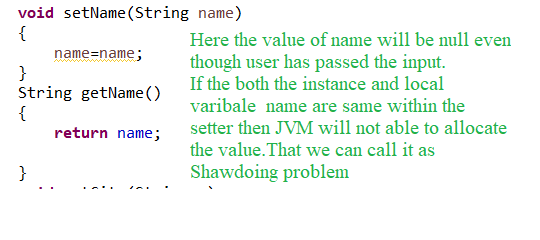
Encapsulation:

Providing controlled access to data members(make data members as private and using getters and setters) by avoiding the Direct Access.

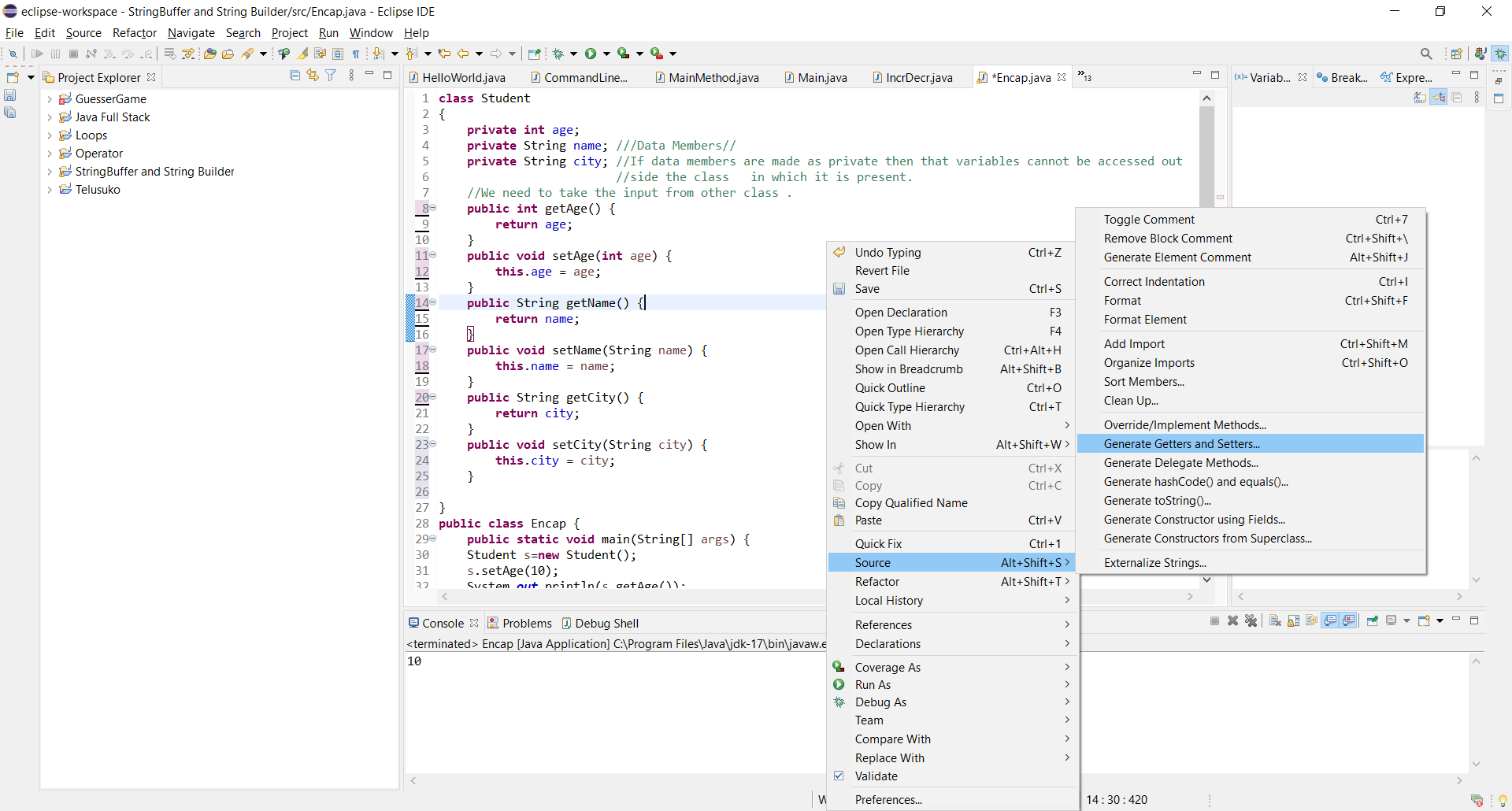
Bean: If all the data members are private then that class will be

Shadowing problem :

If both the instance variable and the local variable name is same then JVM not assign any default value rather than the value provided. (null—for String)—If this key word is not present.



To resolve the issue we can use the this keyword.



Once we define the data members we can define the getters/Setters manually or in the above format as well.

**class** Student

{

**private** **int** age;

**private** String name; ///Data Members//

**private** String city; //If data members are made as private then that variables cannot be accessed out

//side the class in which it is present.

//We need to take the input from other class .

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

}

**public** **class** Encap {

**public** **static** **void** main(String[] args) {

Student s=**new** Student();

s.setAge(10);

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

}

}