creating method with void:

**public** **class** voidmethods {

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

*firstcode*();

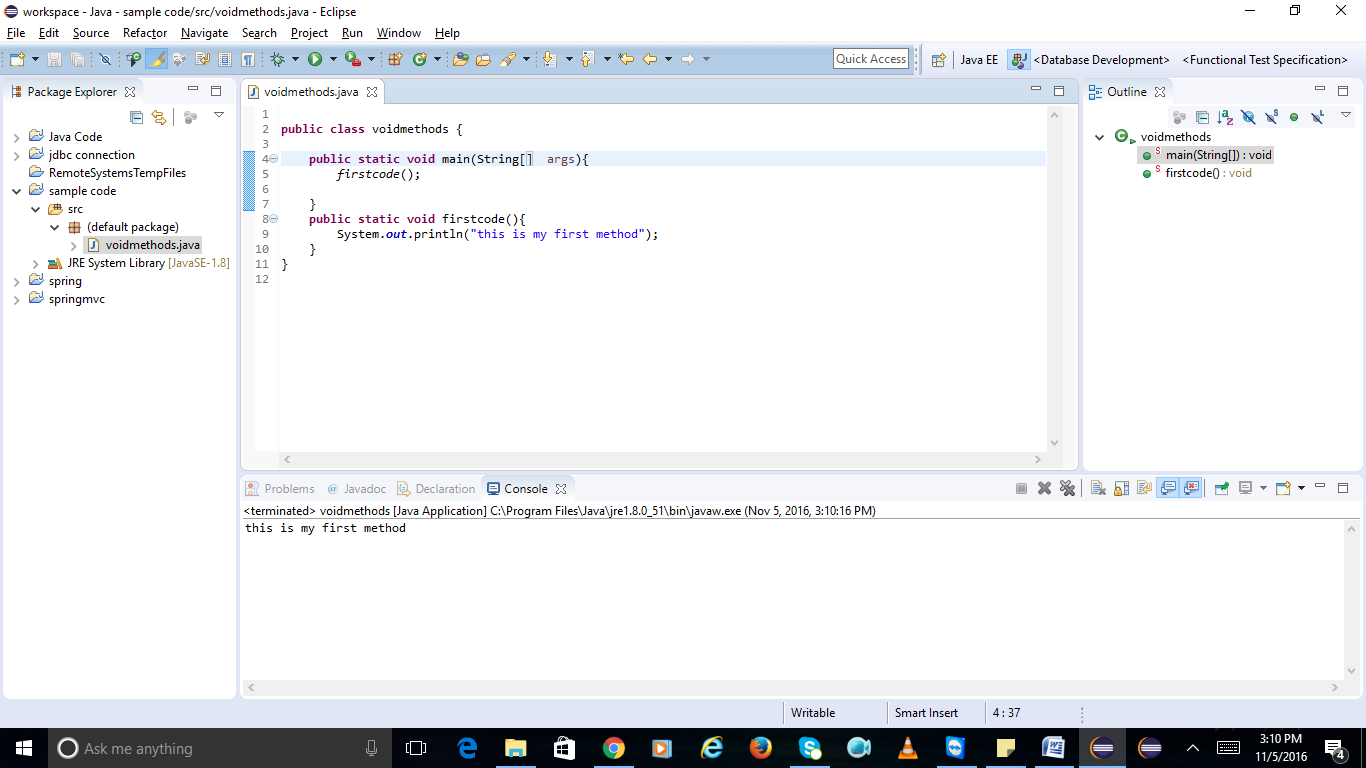
}

**public** **static** **void** firstcode(){

System.***out***.println("this is my first method");

}

}



creating method with return data type:

**public** **class** returnmethod {

**static** **int** *num1*=10;

**static** **int** *num2*=60;

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

System.***out***.println("the answer is " + *getAnswer*());

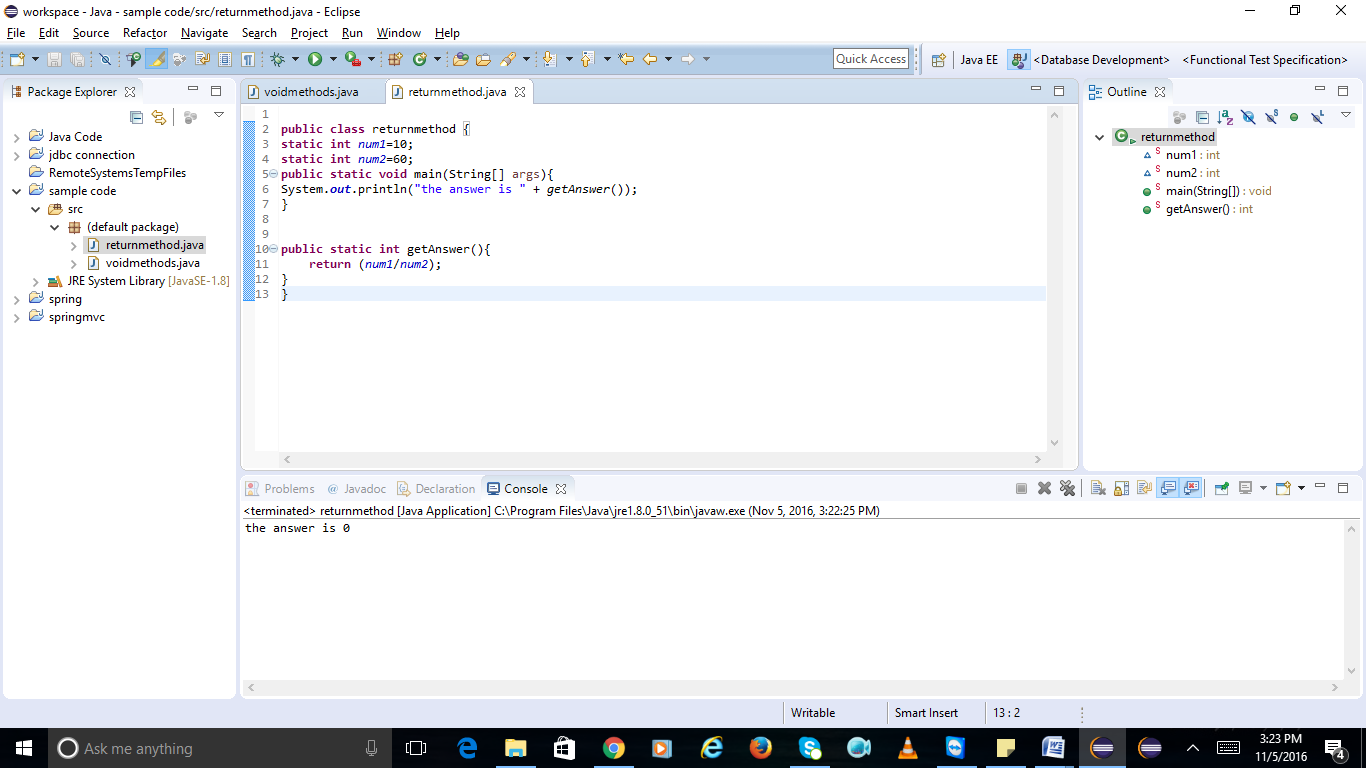
}

**public** **static** **int** getAnswer(){

**return** (*num1*/*num2*);

}

}



creating method with return data type and parameter:

**public** **class** parametersreturntype {

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

*sayhello*("swetha");

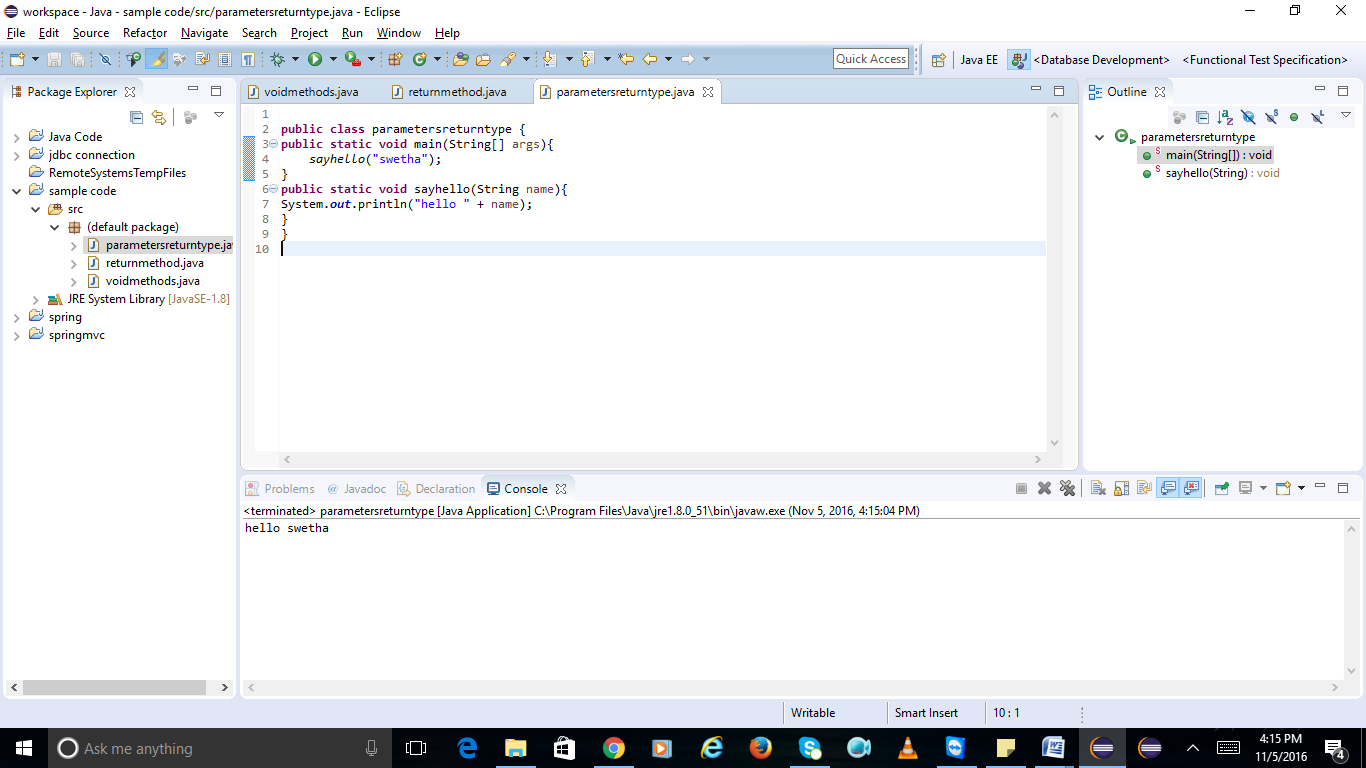
}

**public** **static** **void** sayhello(String name){

System.***out***.println("hello " + name);

}

}



creating variable:

**public** **class** variable {

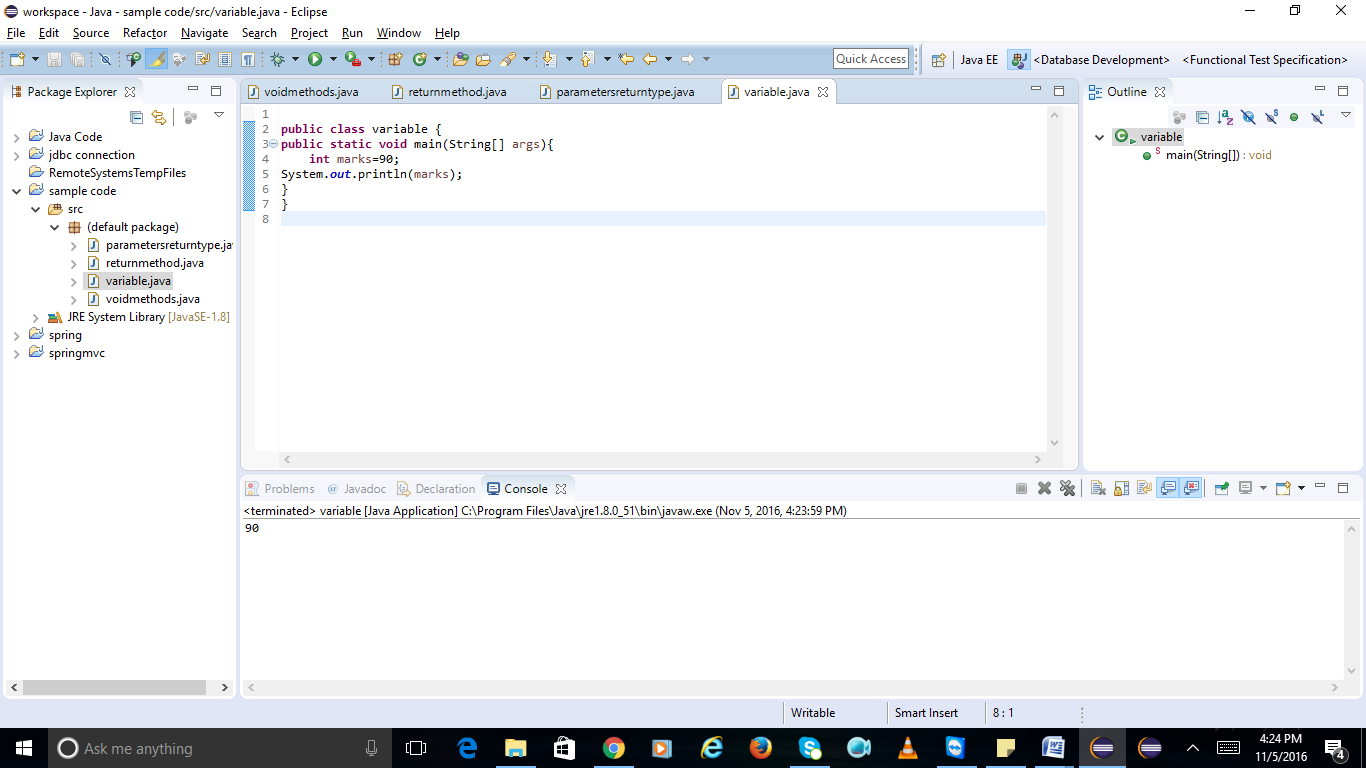
**public** **static** **void** main(String[] args){

**int** marks=90;

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

}

}



creating static property:

**public** **class** staticproperties {

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

Hello hello=**new** Hello();

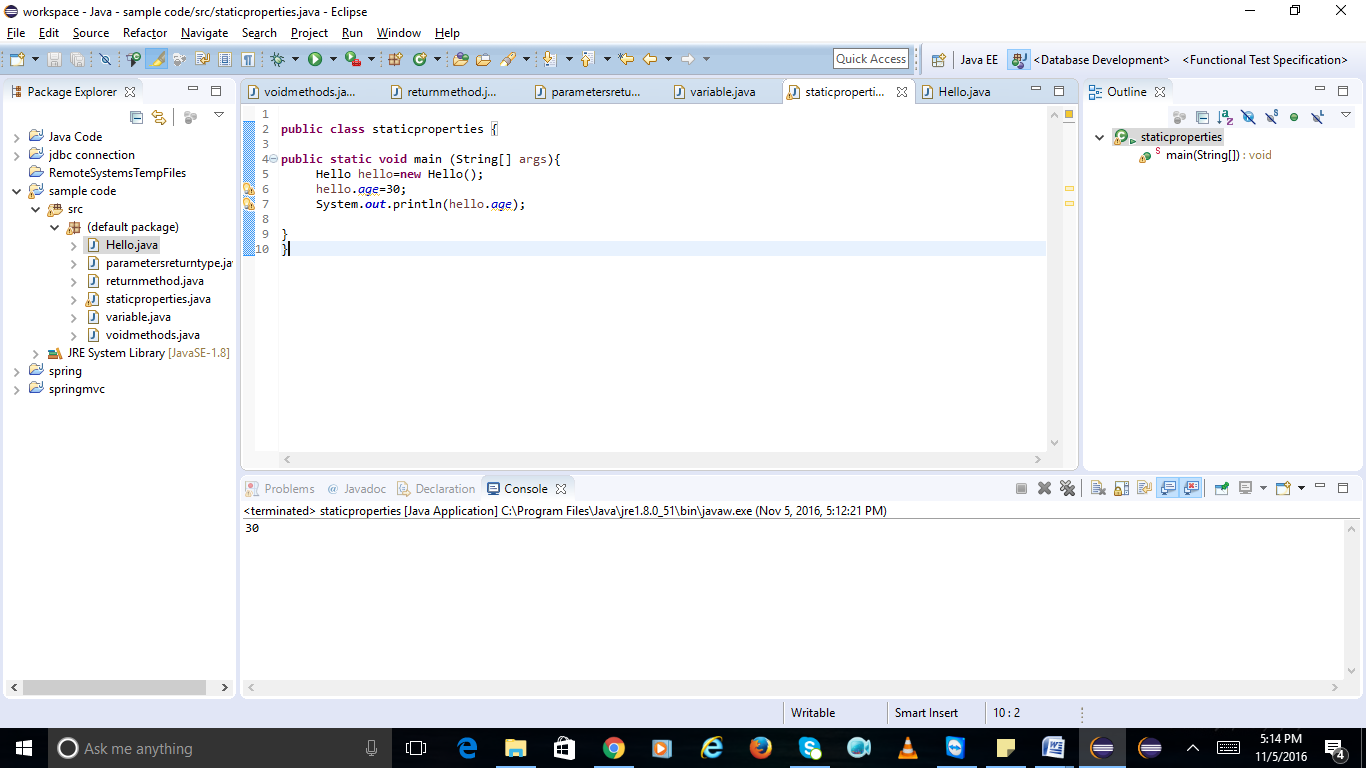
hello.*age*=30;

System.***out***.println(hello.*age*);

**public** **class** Hello {

**public** **static** **int** *age*;

}



Day2:

write code to handle exceptions with try/catch/finally

**public** **class** Exceptions {

**public** **static** **int** retInt(){

**int** a=100;

**try**{

**return** a;

} **catch** (ArithmeticException e){

System.***out***.println("catch called");

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

**return** a;

}**finally**{

a=5000;

System.***out***.println("finally called");

**return** a;

}

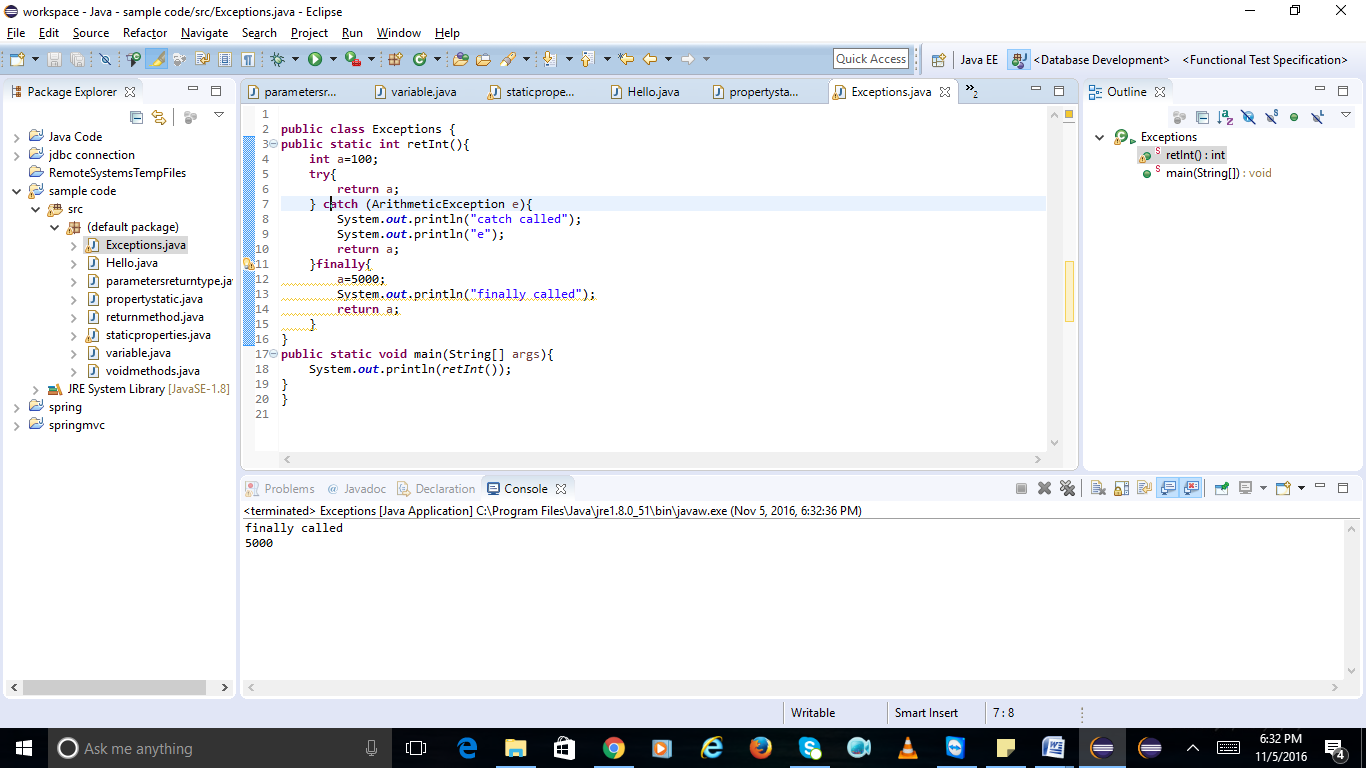
}

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

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

}

}



what is final keyword

**public** **class** tuna {

**private** **int** sum;

**private** **final** **int** NUMBER;

**public** tuna (**int** x){

NUMBER=x;

}

**public** **void** add(){

sum+=NUMBER;

}

**public** String toString(){

**return** String.*format*("sum =%d/n",sum);

}

}

**public** **class** apples {

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

tuna tuna=**new** tuna(10);

**for**(**int** i=0;i<5;i++){

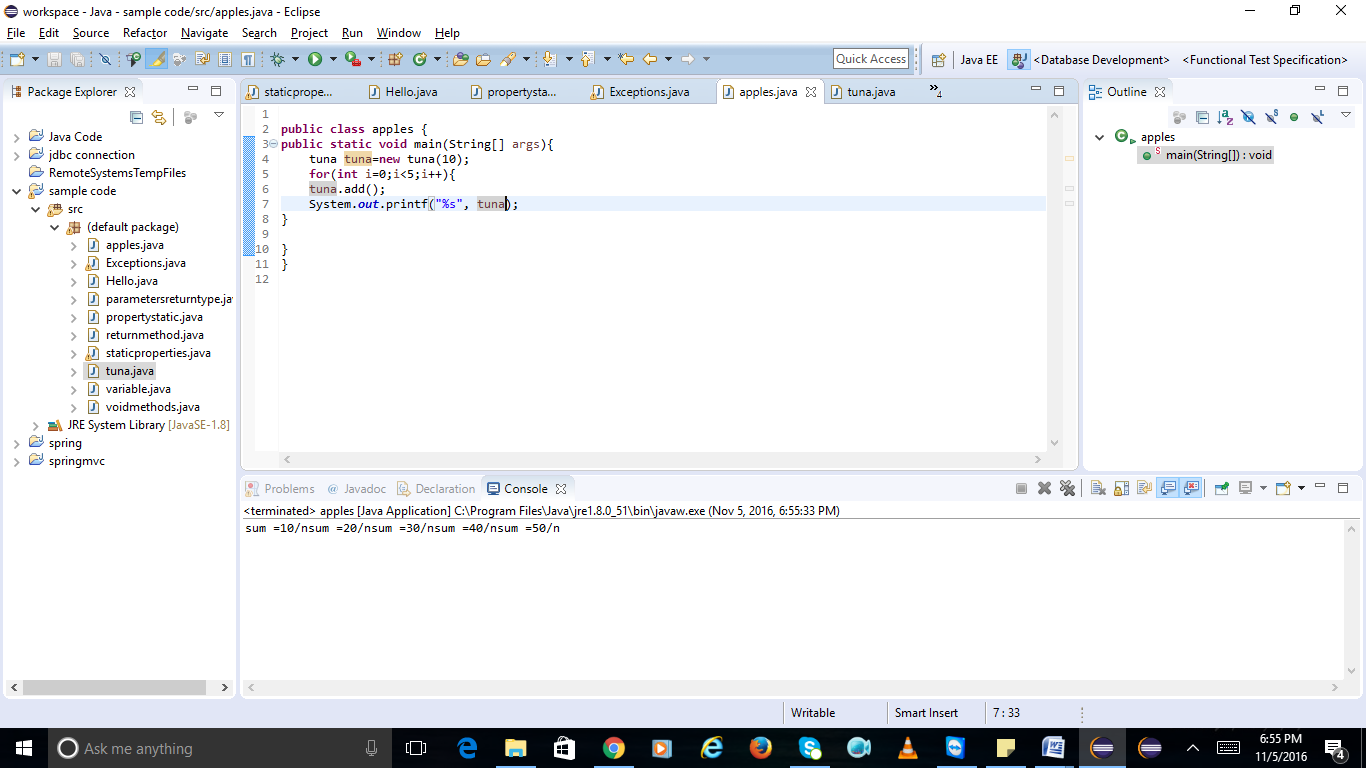
tuna.add();

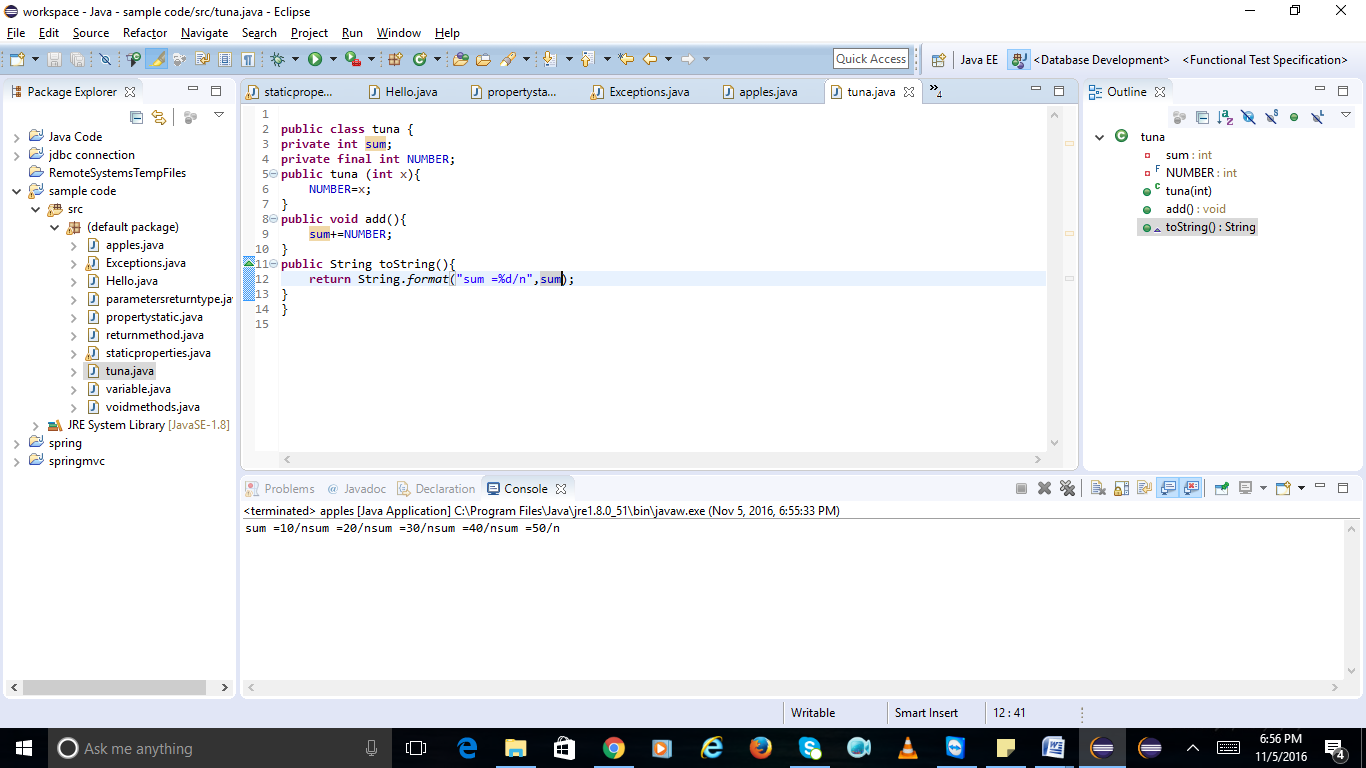
System.***out***.printf("%s", tuna);

}

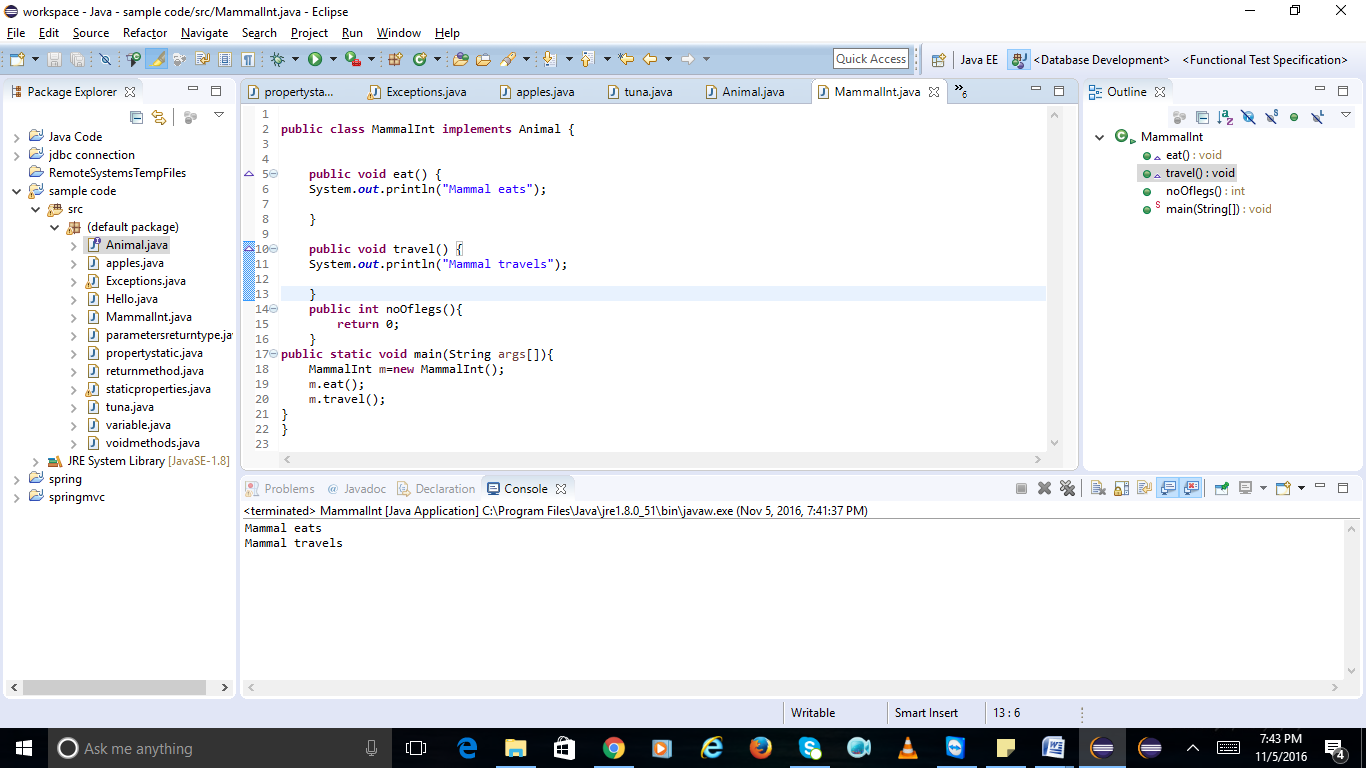
}

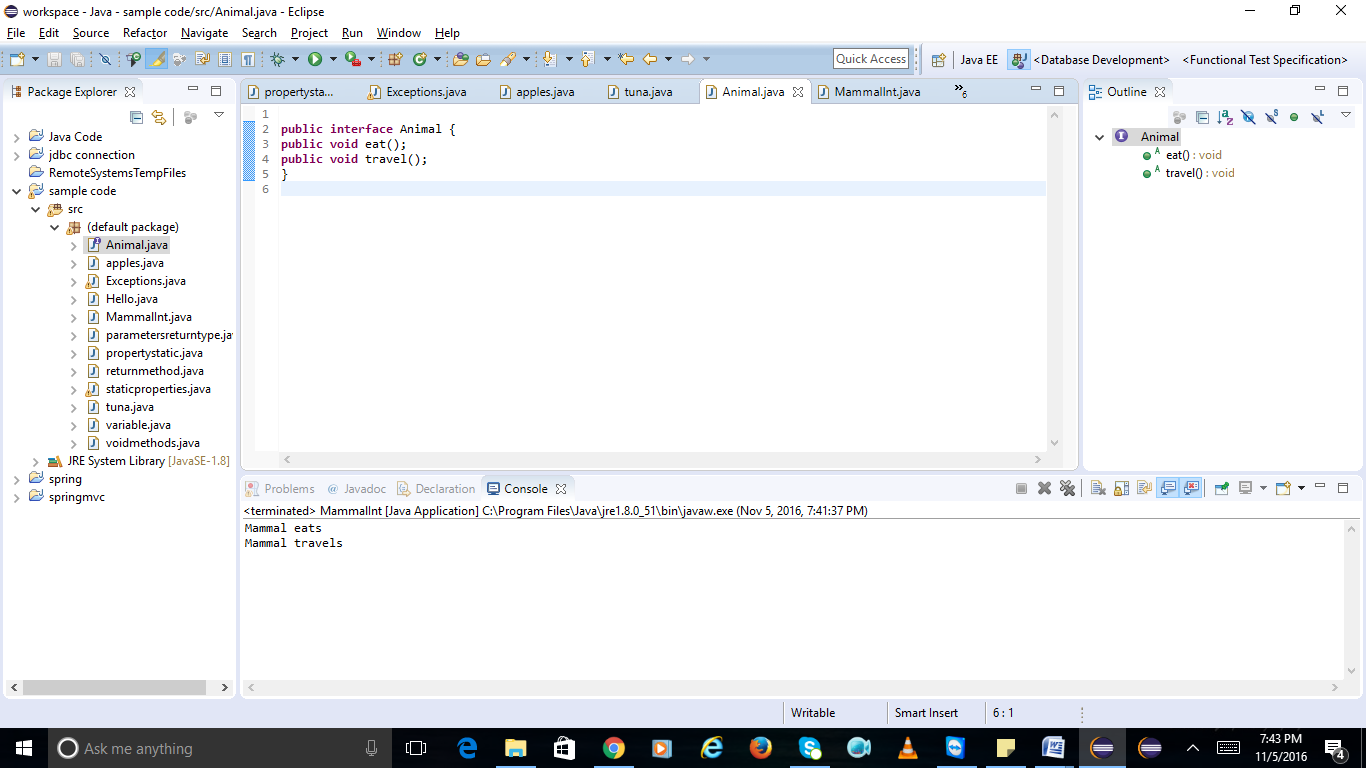
}





write code for interface and create class to implement that interface





**public** **class** MammalInt **implements** Animal {

**public** **void** eat() {

System.***out***.println("Mammal eats");

}

**public** **void** travel() {

System.***out***.println("Mammal travels");

}

**public** **int** noOflegs(){

**return** 0;

}

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

MammalInt m=**new** MammalInt();

m.eat();

m.travel();

}

}

write code for creating abstract class:

**public** **abstract** **class** Employee {

**private** String name;

**private** String address;

**private** **int** number;

**public** Employee(String name, String address, **int** number) {

System.***out***.println("Constructing an Employee");

**this**.name = name;

**this**.address = address;

**this**.number = number;

}

**public** **double** computePay() {

System.***out***.println("Inside Employee computePay");

**return** 0.0;

}

**public** **void** mailCheck() {

System.***out***.println("Mailing a check to " + **this**.name + " " + **this**.address);

}

**public** String toString() {

**return** name + " " + address + " " + number;

}

**public** String getName() {

**return** name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String newAddress) {

address = newAddress;

}

**public** **int** getNumber() {

**return** number;

}

}

**public** **class** AbstractDemo {

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

Salary s = **new** Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);

Employee e = **new** Salary("John Adams", "Boston, MA", 2, 2400.00);

System.***out***.println("Call mailCheck using Salary reference --");

s.mailCheck();

System.***out***.println("\n Call mailCheck using Employee reference--");

e.mailCheck();

}

}

**public** **class** Salary **extends** Employee

{

**private** **double** salary; // Annual salary

**public** Salary(String name, String address, **int** number, **double** salary) {

**super**(name, address, number);

setSalary(salary);

}

**public** **void** mailCheck() {

System.***out***.println("Within mailCheck of Salary class ");

System.***out***.println("Mailing check to " + getName() + " with salary " + salary);

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** newSalary) {

**if**(newSalary >= 0.0) {

salary = newSalary;

}

}

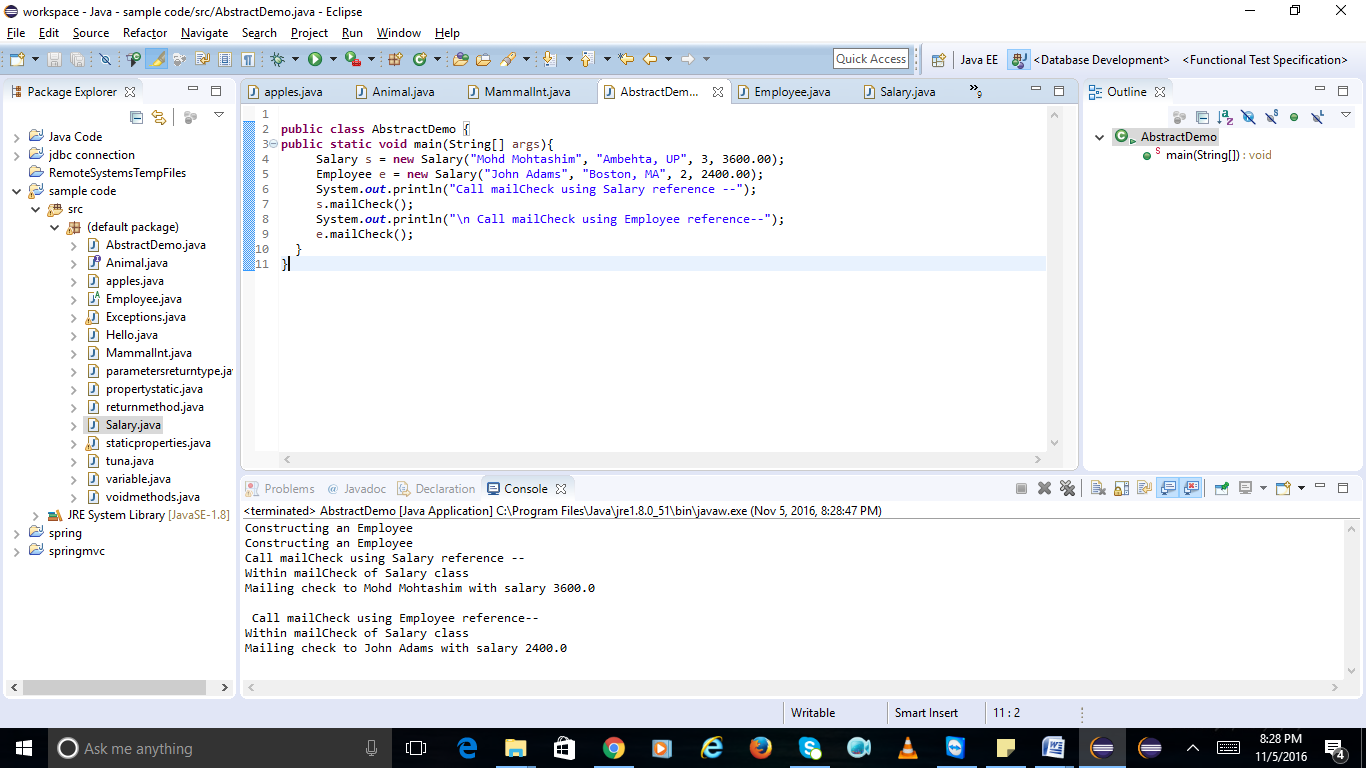
**public** **double** computePay() {

System.***out***.println("Computing salary pay for " + getName());

**return** salary/52;

}

}



implement method overriding

**class** Animal1 {

**public** **void** move() {

System.***out***.println("Animals can move");

}

}

**class** Dog **extends** Animal1 {

**public** **void** move() {

System.***out***.println("Dogs can walk and run");

}

}

**public** **class** TestDog {

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

Animal1 a = **new** Animal1(); // Animal reference and object

Animal1 b = **new** Dog(); // Animal reference but Dog object

a.move(); // runs the method in Animal class

b.move(); // runs the method in Dog class

}

}

