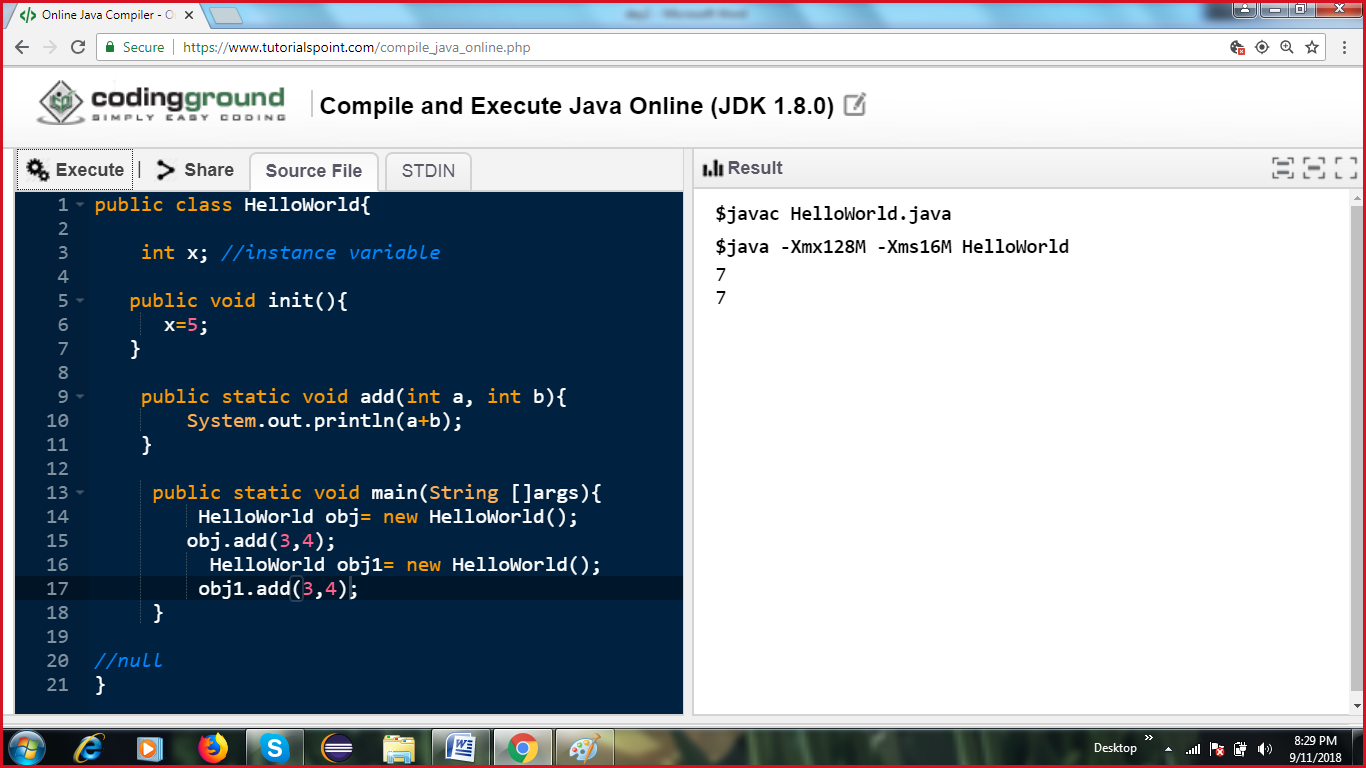
Access modifiers:

1. Private: only within the same class
2. Public: by all
3. Default: within the same package
4. Protected: within the same package or child classes in a diff package

Class: public, default

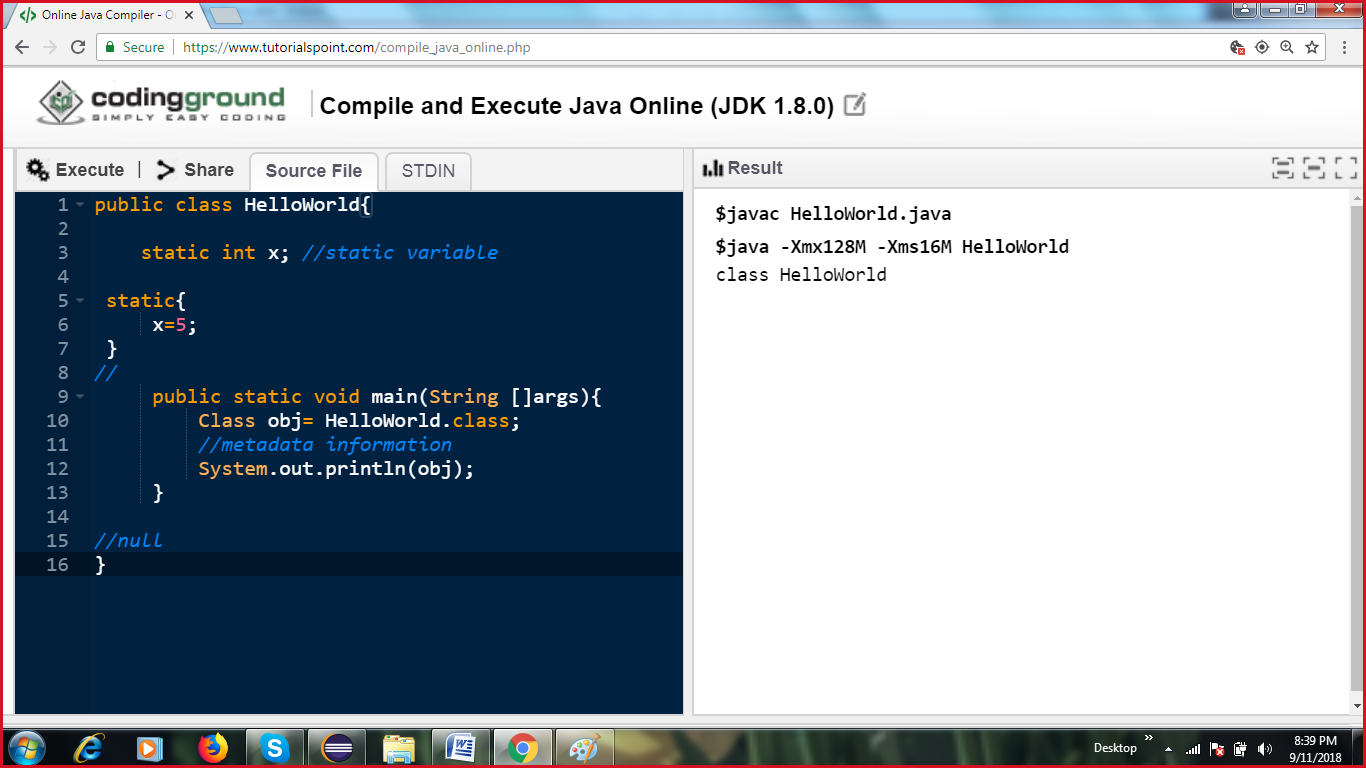
Modifiers:

1. Final
2. Class: Cannot be inherited
3. Method: cannot be overridden
4. Variable: cannot re-assign the variable
5. Static
6. Variable: method area/perm gen space
7. Method
8. block



Instance block is called when an object is created, before the constructor is invoked

Static block is called once when your class is loaded



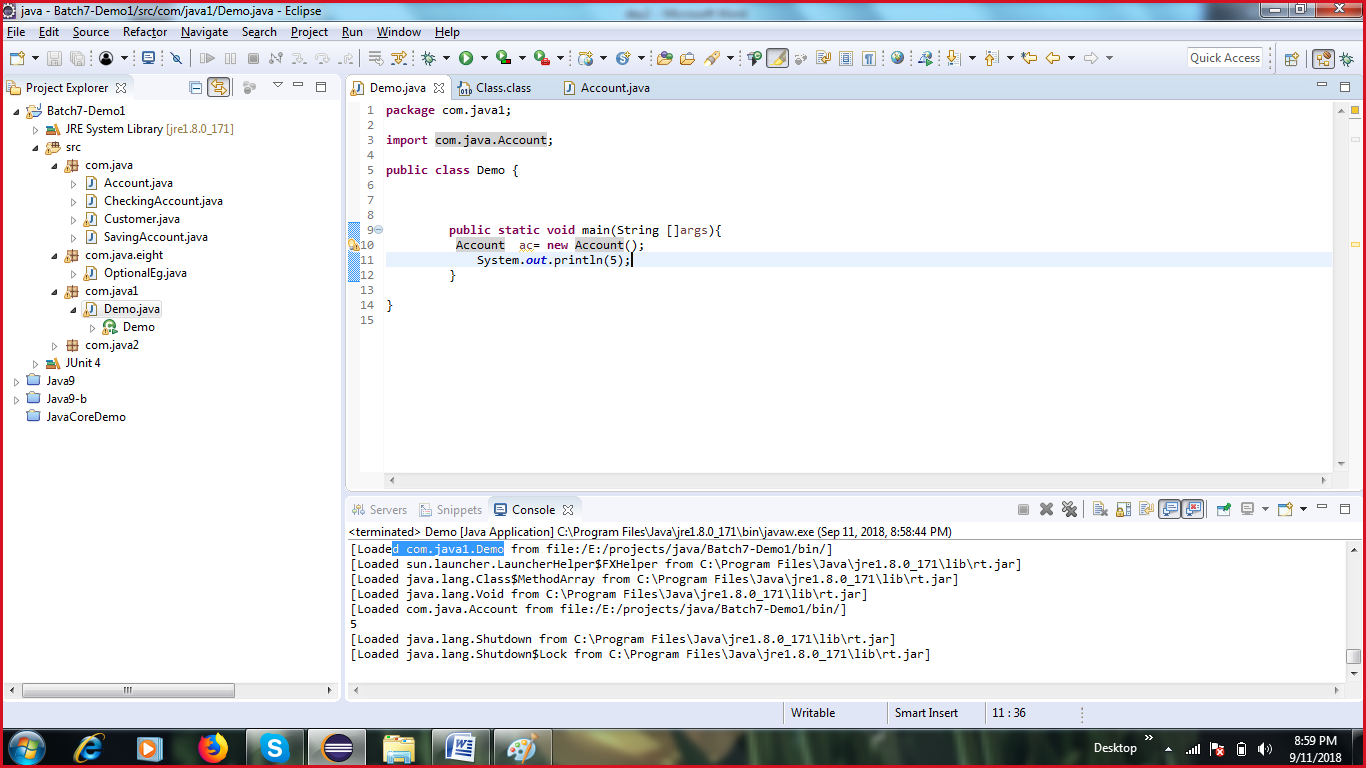
Class Loading:

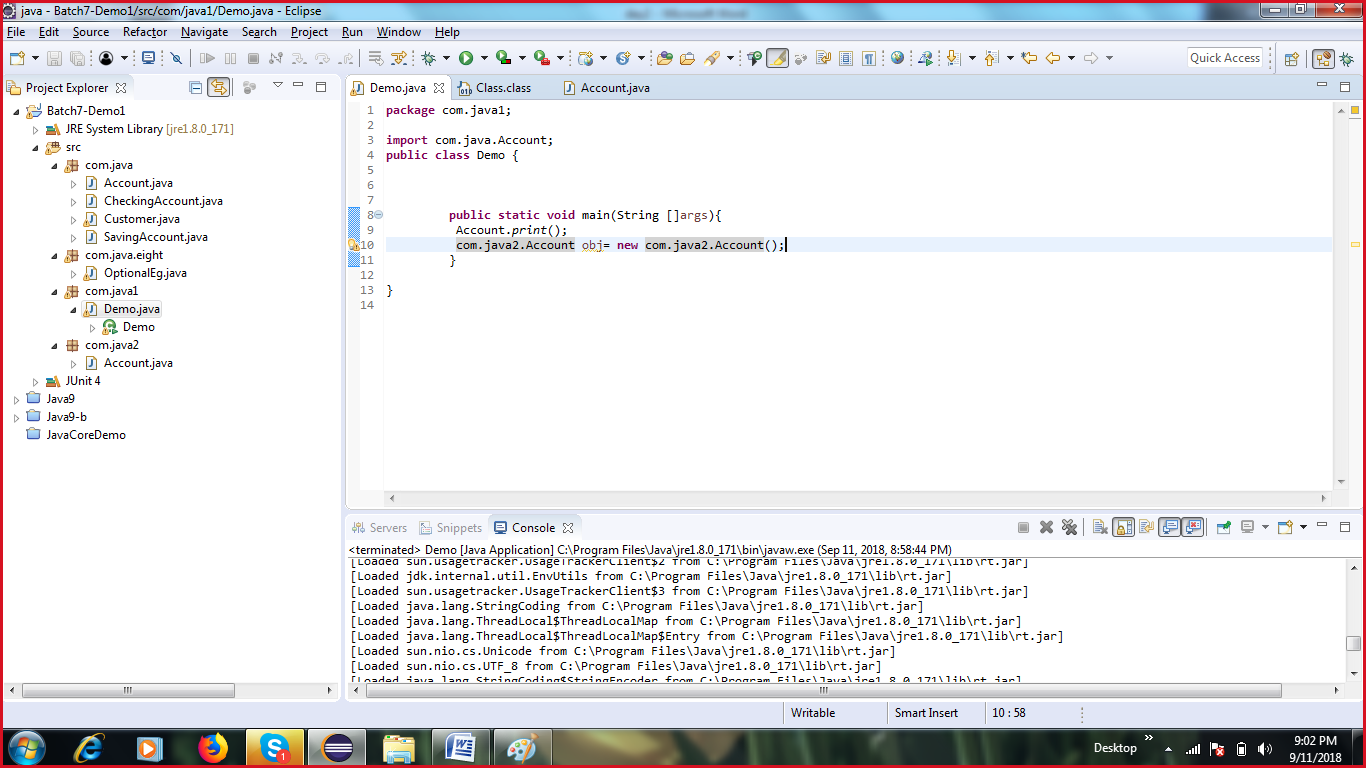
Class Loader: 3 types of classloaders

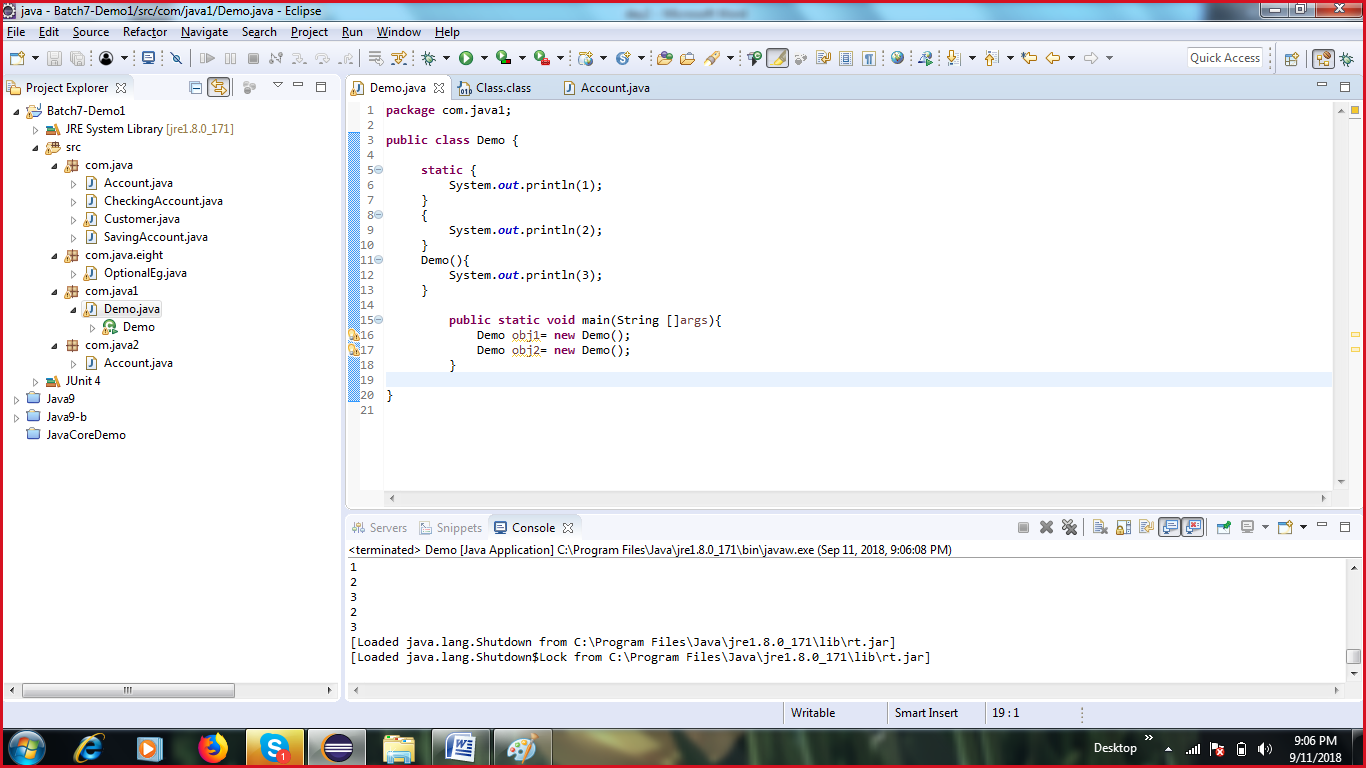
1. Bootstrap: Loads classes from rt.jar
2. Extension: loads classes from the ext folder
3. System/Application: Loaded classes from classpath

Delegating class loading pattern?

1. Load all the classes present in rt.jar

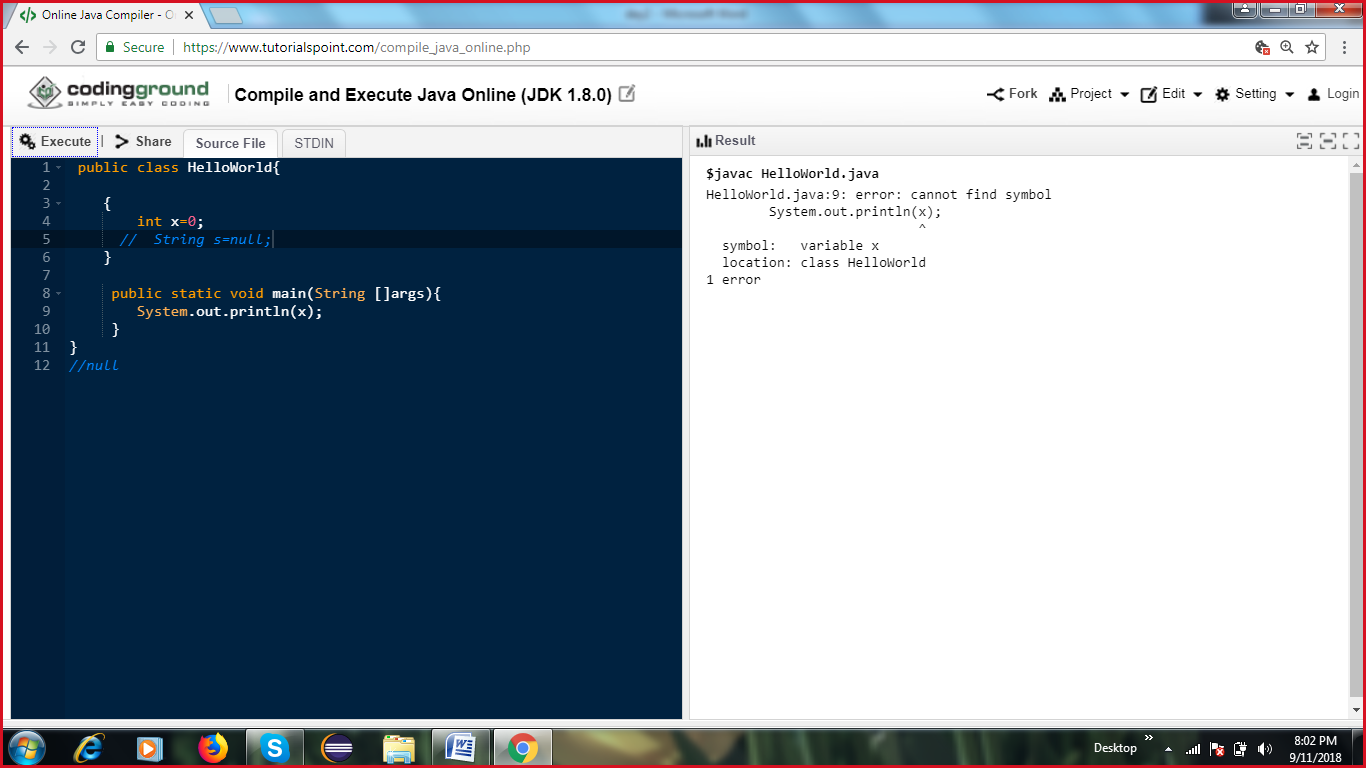


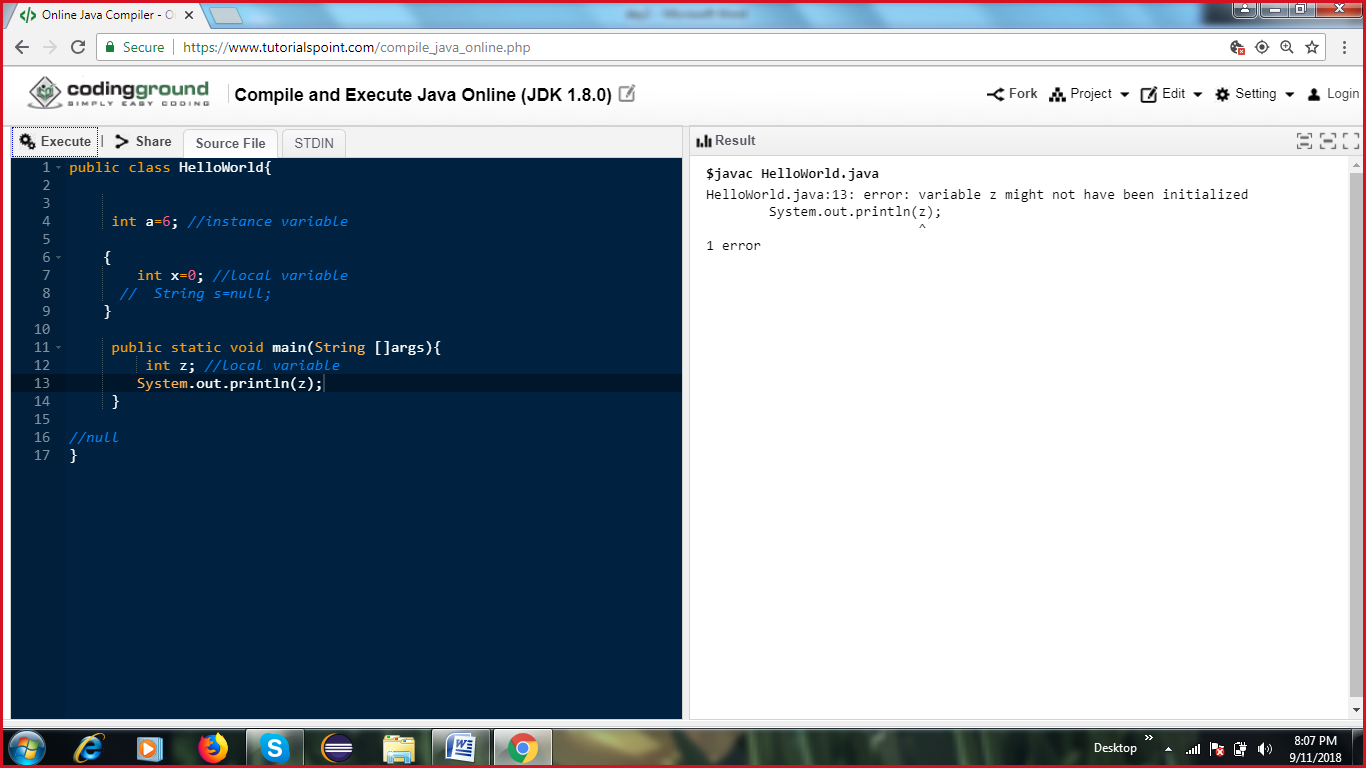


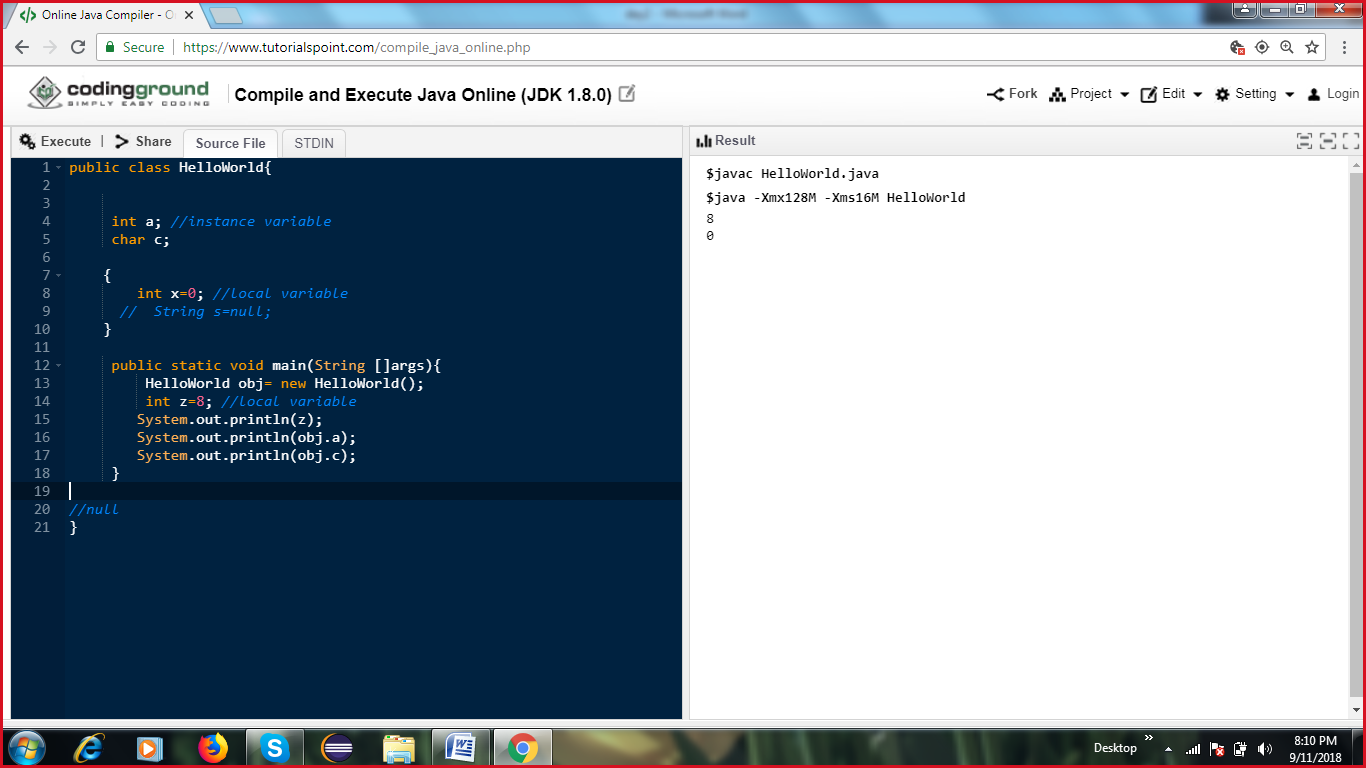


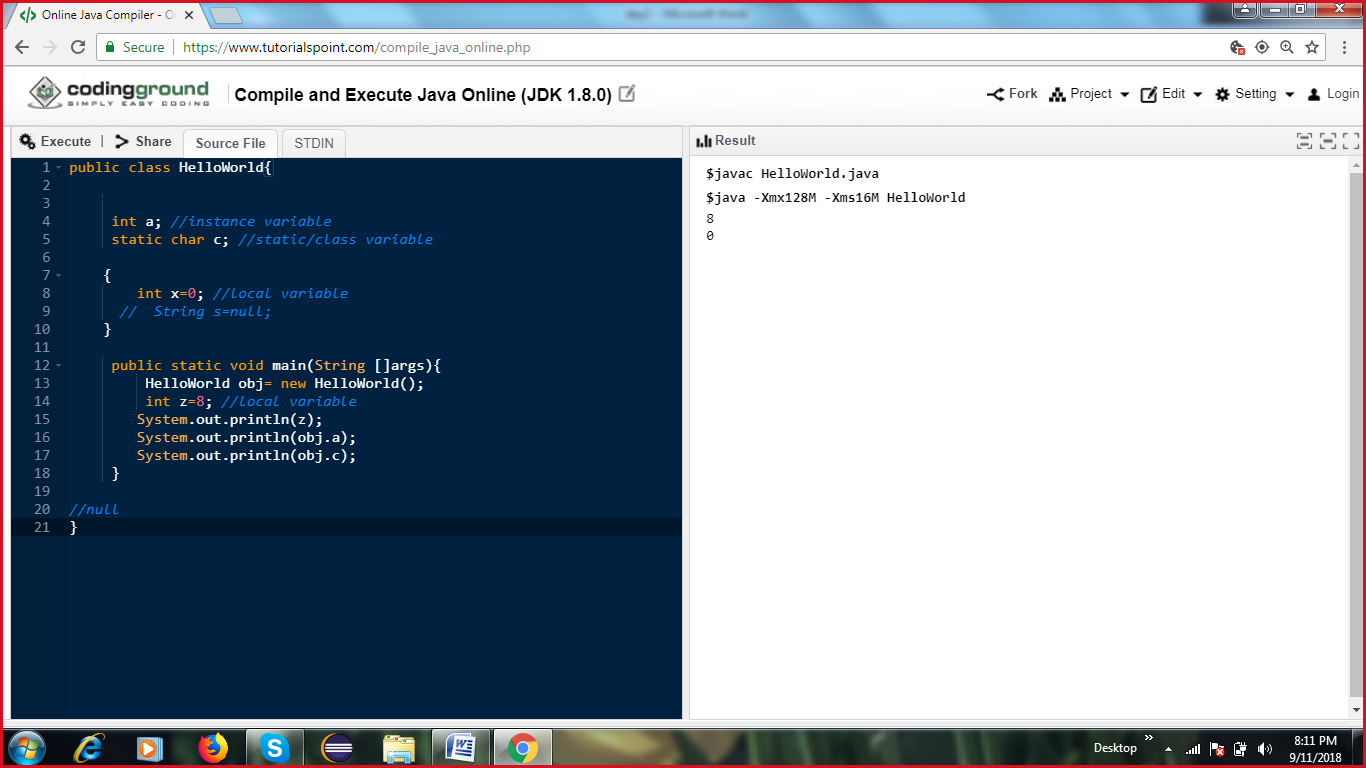
Variables:

1. Local: defined within a method or a block. They should be assigned a value before we use them
2. Instance: They belong to a particular object. They are initialize by default
3. Static/Class: . They are initialize by default

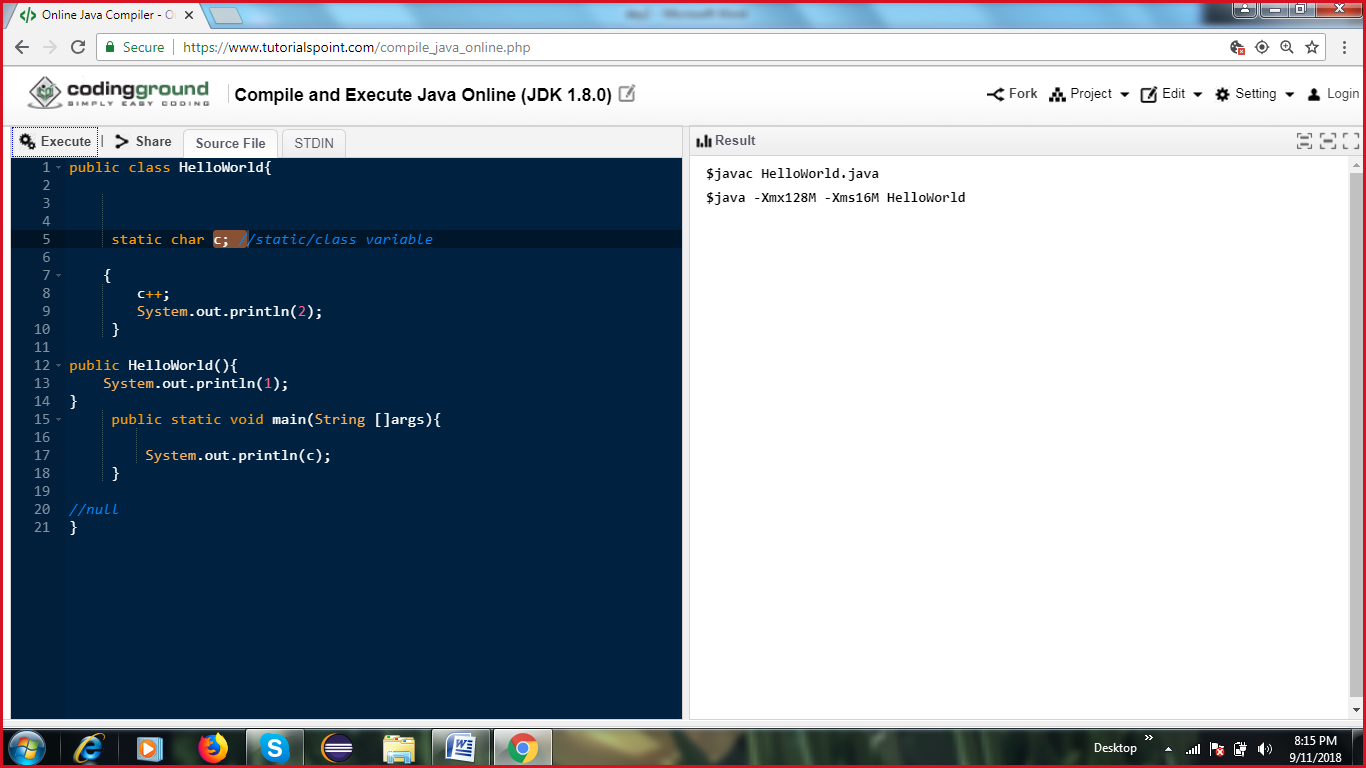


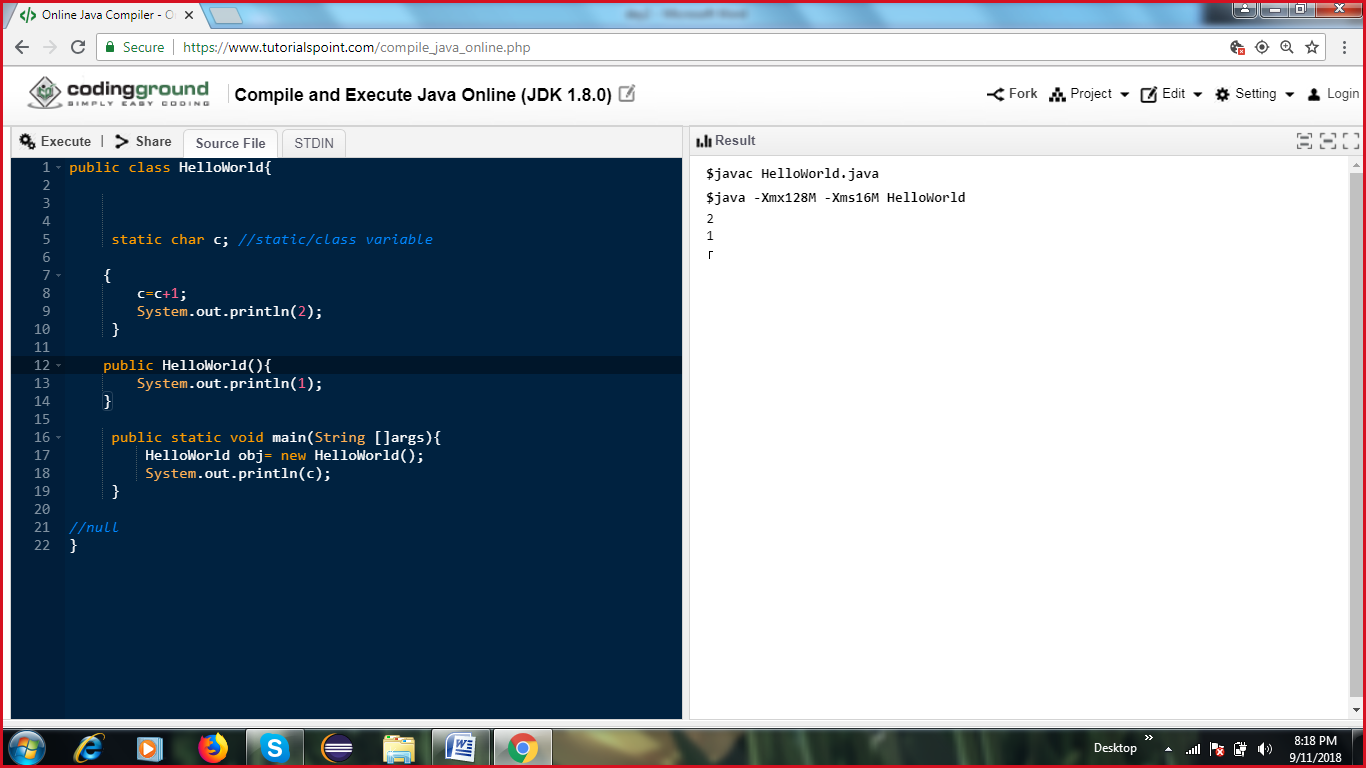


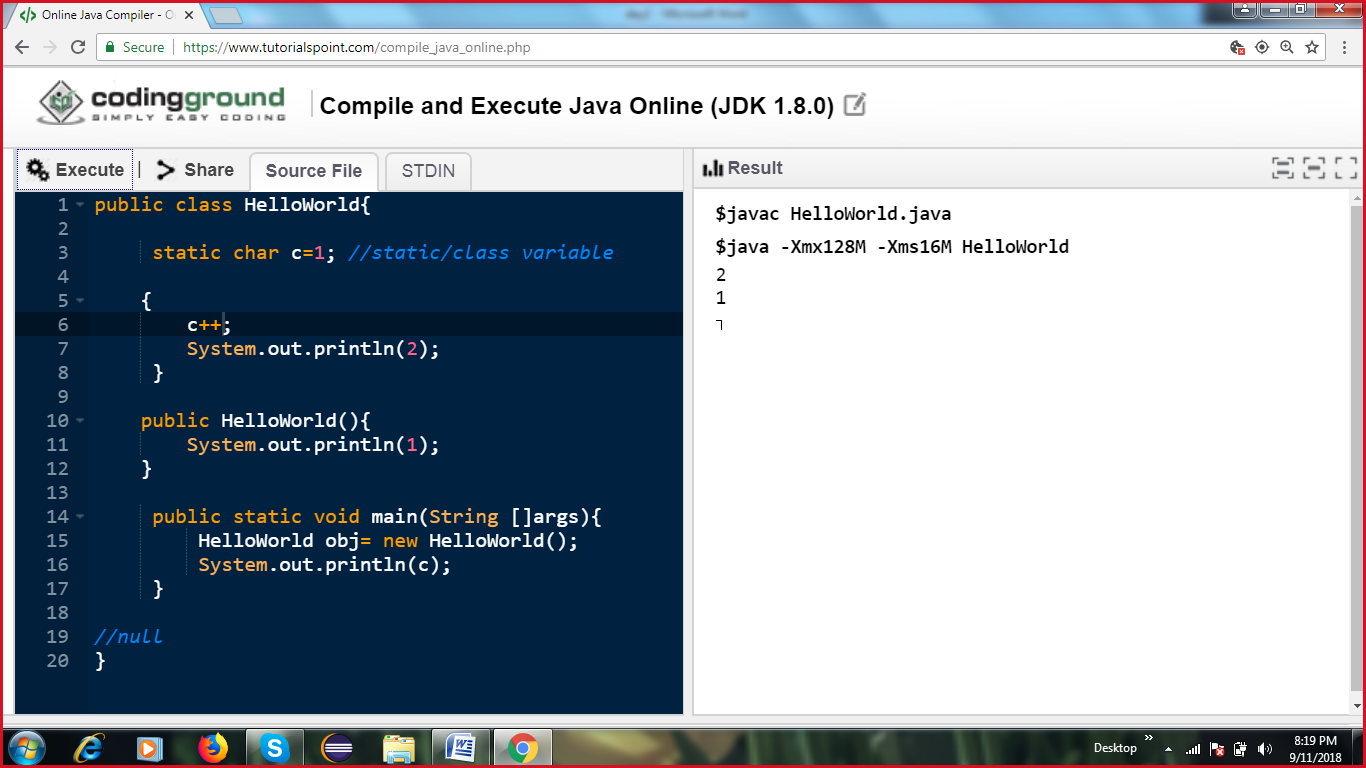


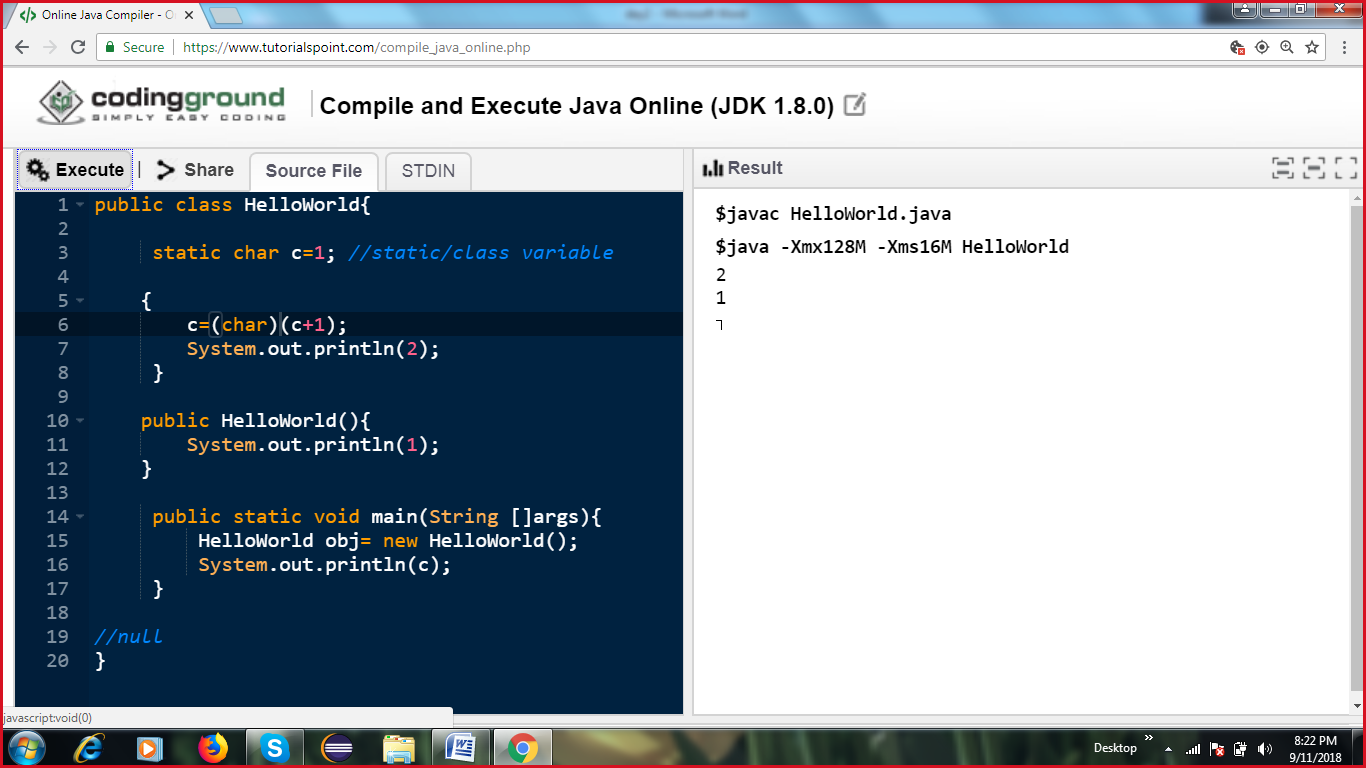


}



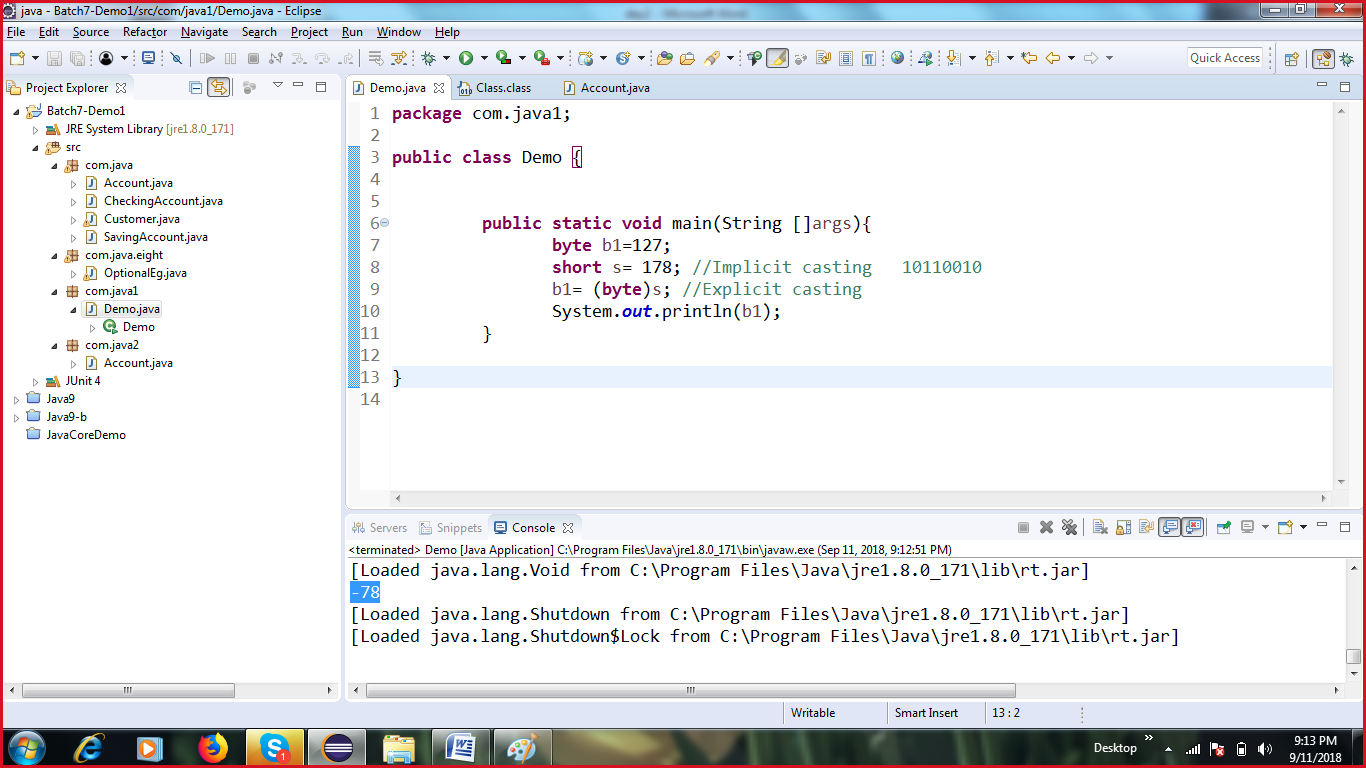






Implict casting

Byte | short | int | long | float| double | char



Explicit casting

byte -> short -> int -> long -> float -> double (implicit)

double -> float -> long-> int -> short -> byte (explicit)

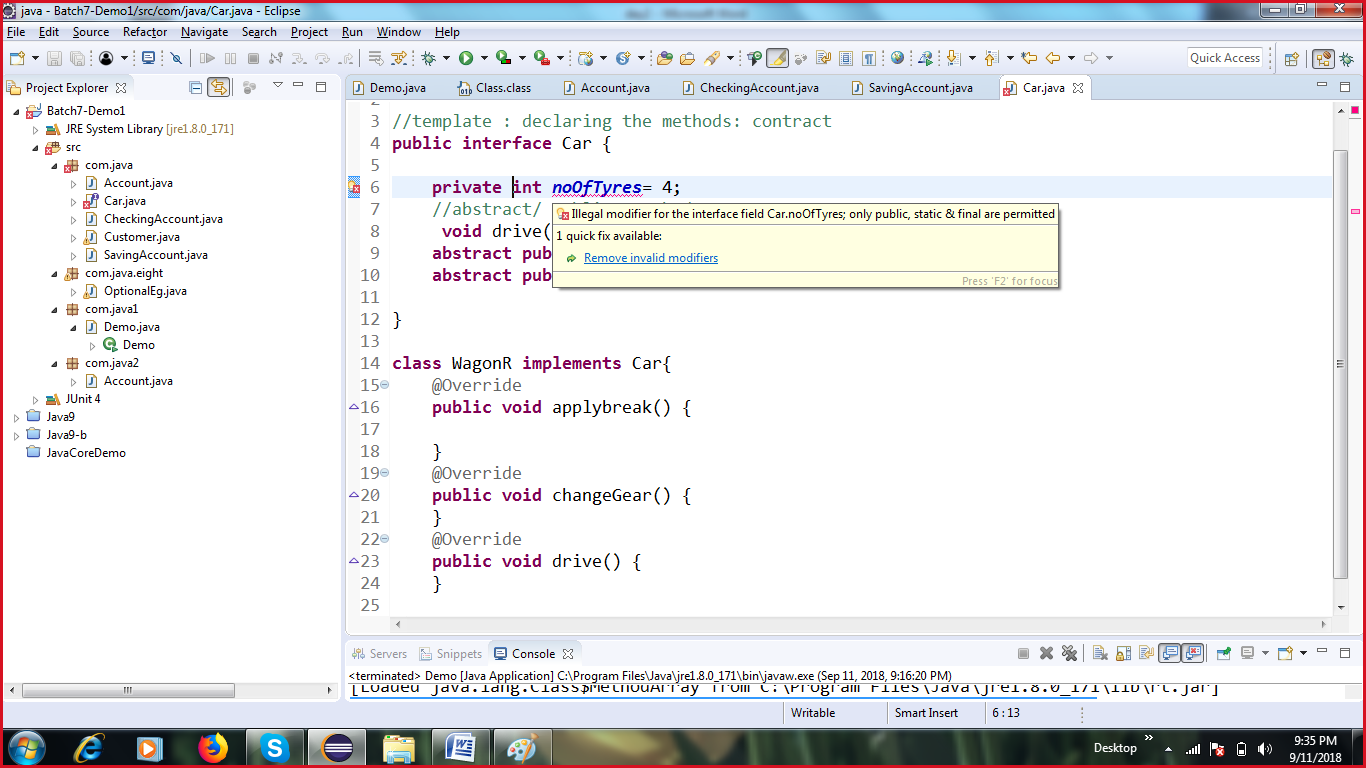
OOPs concepts:

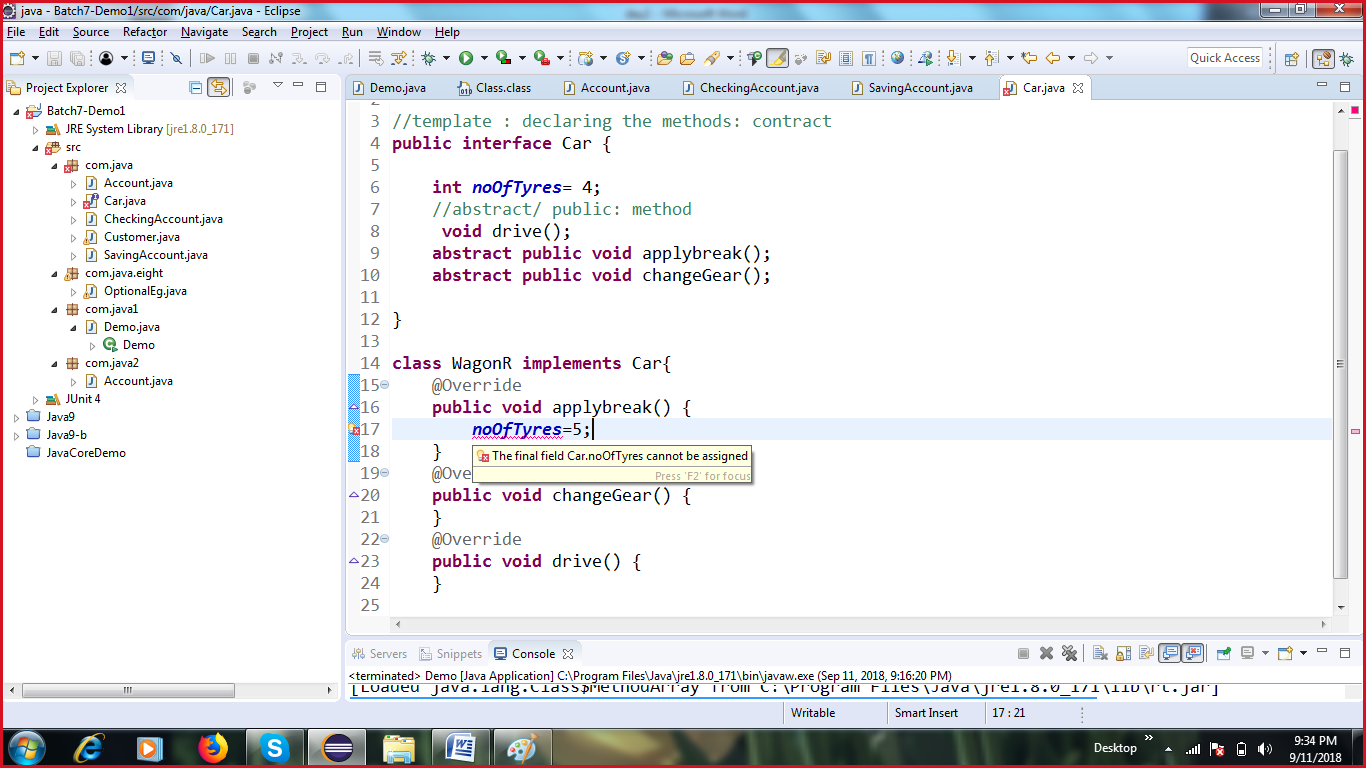
1. Inheritance: Is-A

extends / implements (interfaces)

Methods of an interface are implicitly public and abstract.

Properties in interface are implicitly public, static and final





abstract

1. Method: declaring a method, not providing the implementation for it

public void drive();

1. Class: which has 0 or more abstract methods. If a class/interface has an abstract method, so the child class should either provide the implementation for that abstract method or should be declared to be abstract. You cannot create an object of an abstract class.

**Multi-level Inheritance**

Class A

Class B extends A

Class C extends B

**Multiple Inheritance**

Class D extends A, E (Not supported in java)

One class cannot extend more than 1 class

One class can implement more than 1 interface

Interface Printable{

Void print();

}

Interface Runnable extends Printable{

Void run();

}

Class extends another class

Class implements an interface

Interface extends an interface

Class A{

void m1(){

}

}

Interface I1{

void print();

void m1();

}

abstract Class B extends A implements I1{

Public static void main(String args[]){

//B b= new B();

//b.m1();

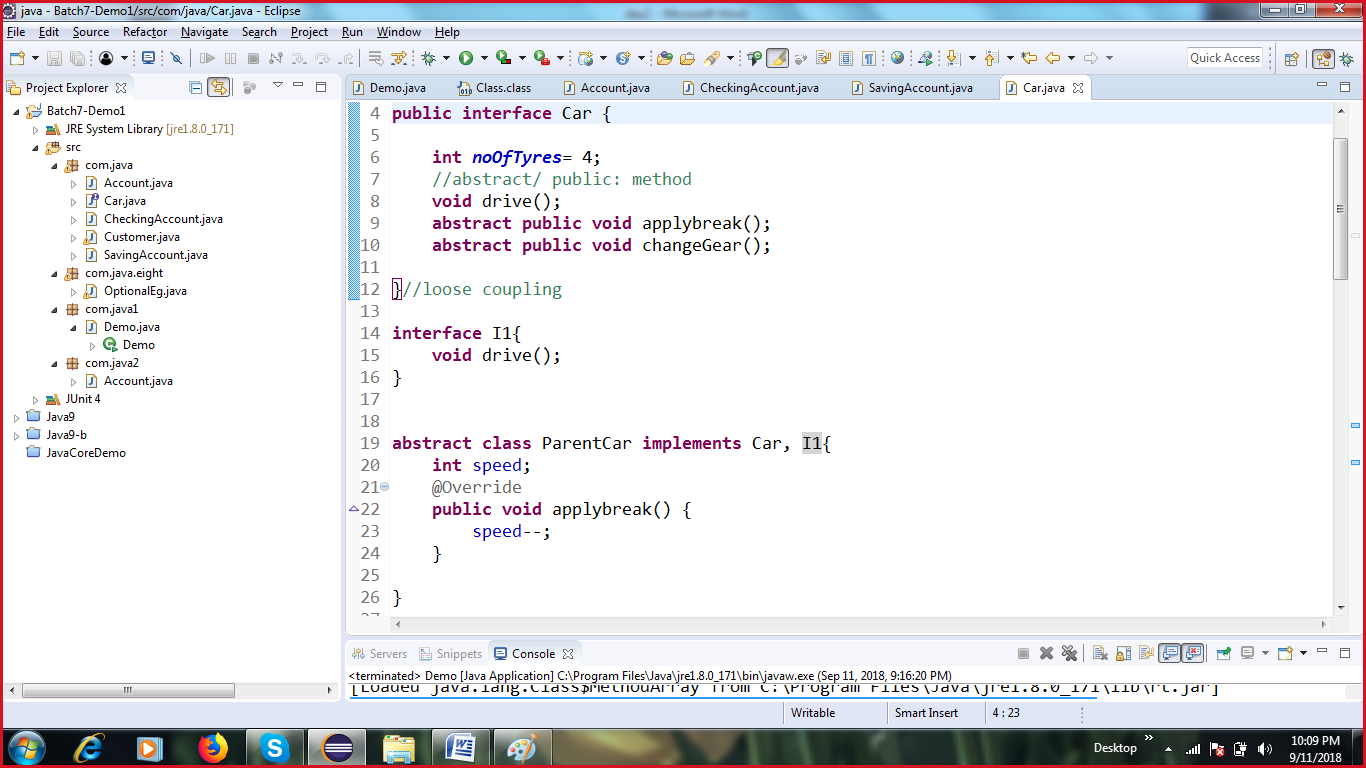
//b.print();

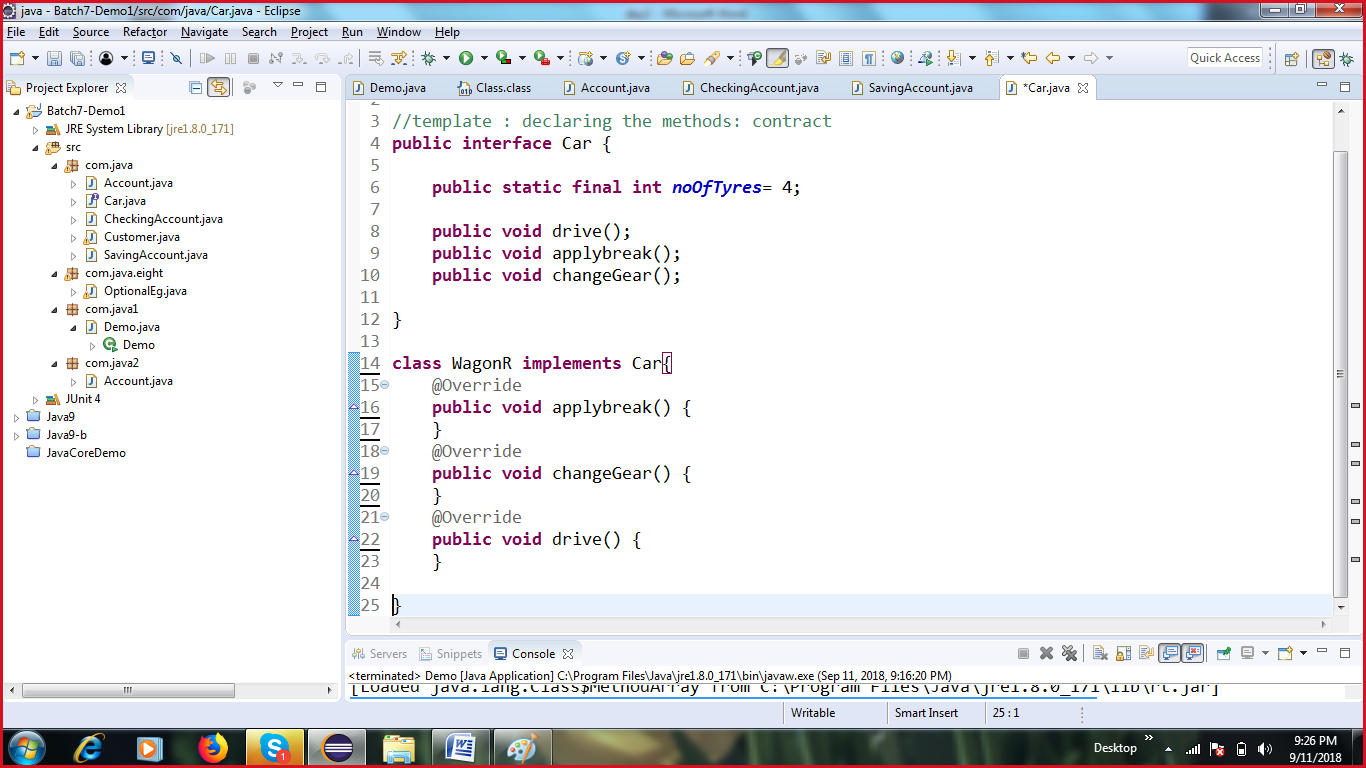
}

//void print(){

//}

}

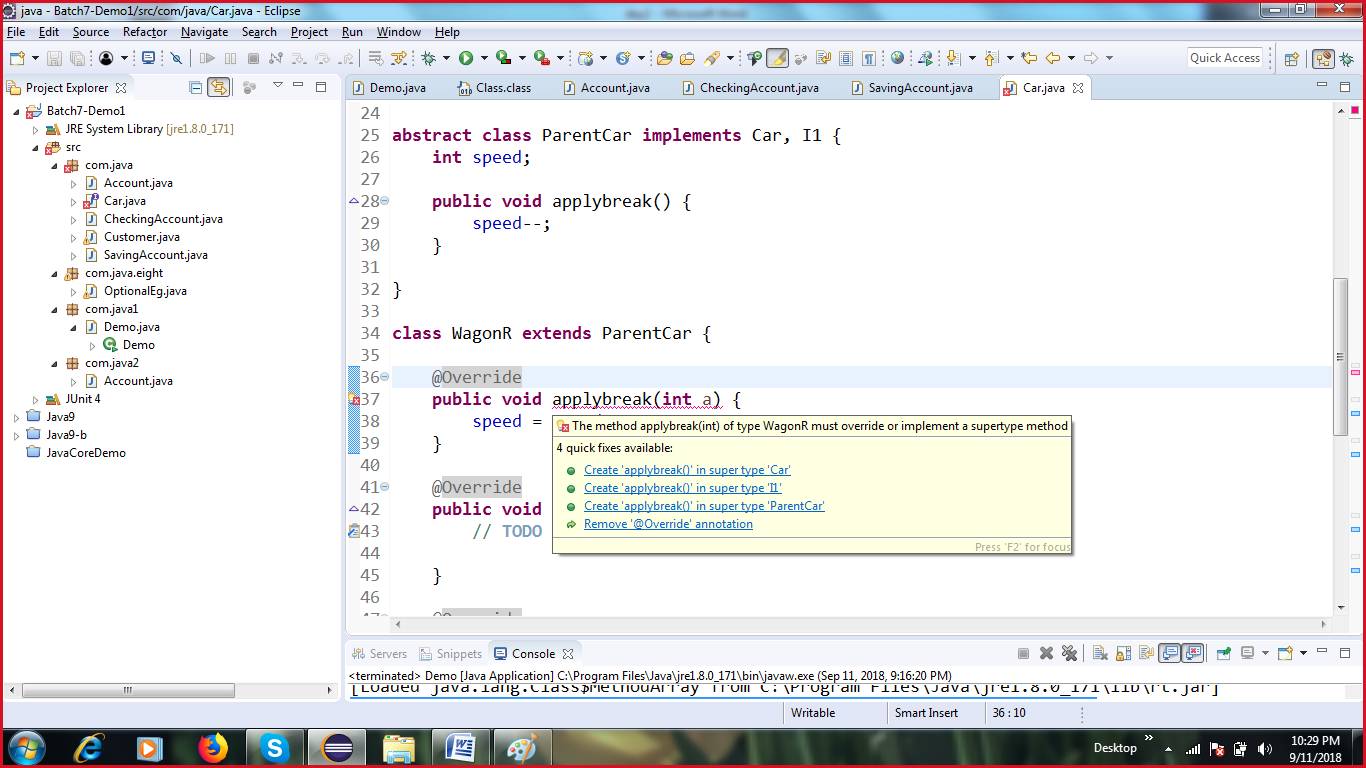




**Method overriding**

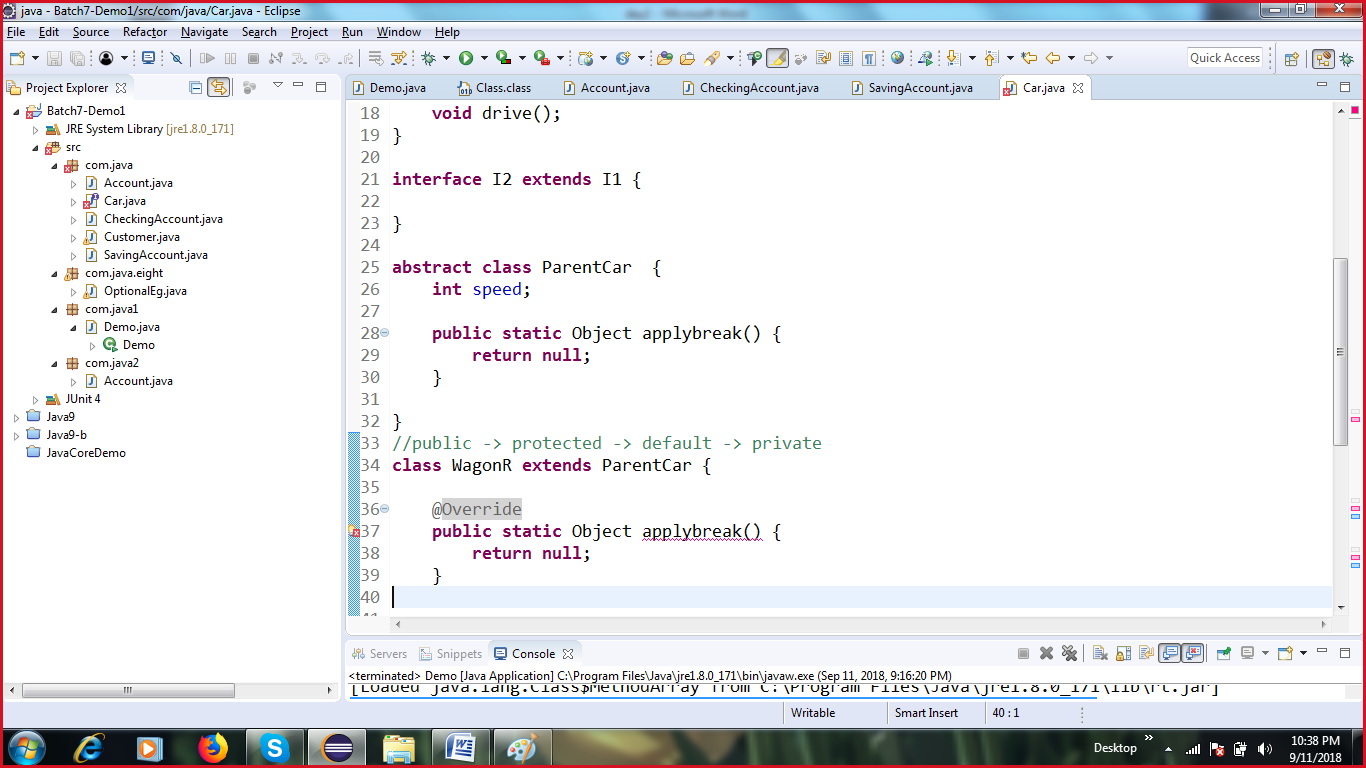
Provide a diff implementation for any of the method in the parent class.

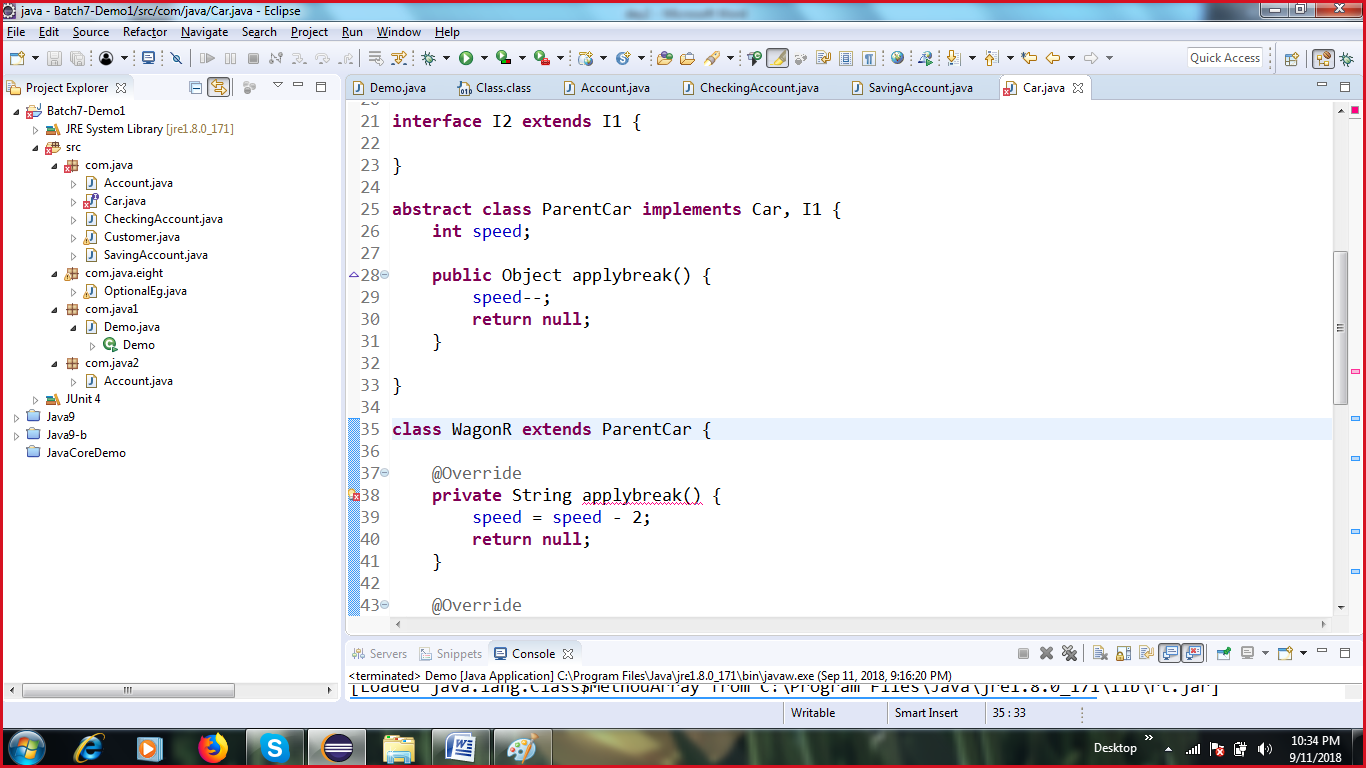
@Override: Compile time check that u have overridden methods properly.

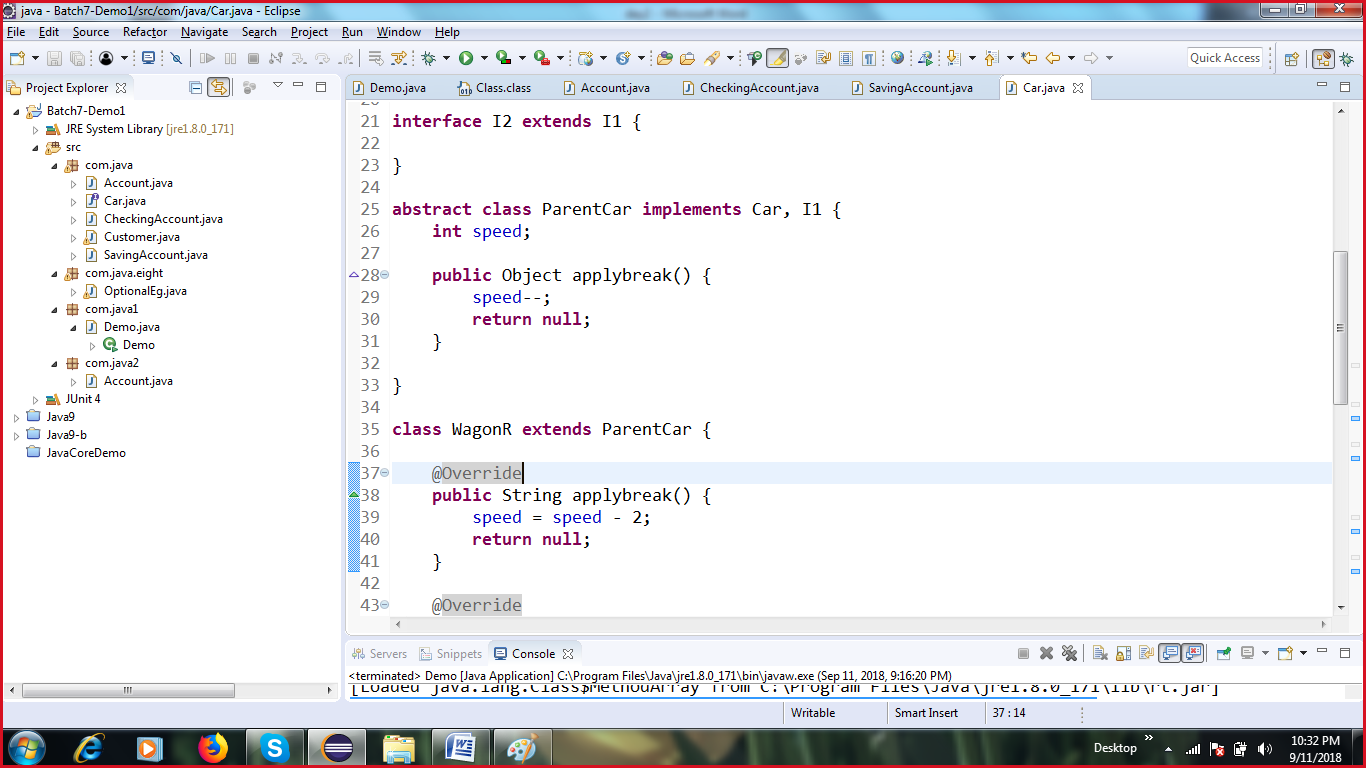


Rules of overriding:

1. Name of the method should be same
2. Arguments should be same
3. Return types should be co-variant
4. Access modifiers: Cannot make it more restrictive. Private cannot be overridden.
5. Static methods cannot be overridden



1. 



**Method overloading:**

1. Can be done in the same class
2. Method names shud be same
3. Arguments shud be diff: type/order/ number
4. Return type: does not matter
5. Access modifier: does not matter

