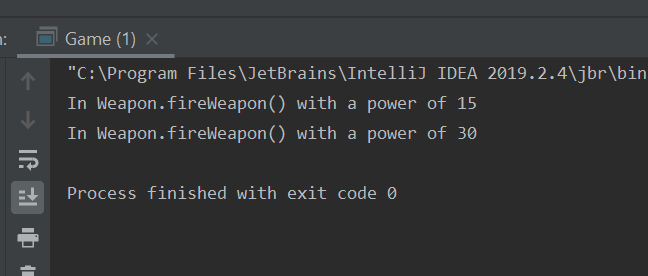
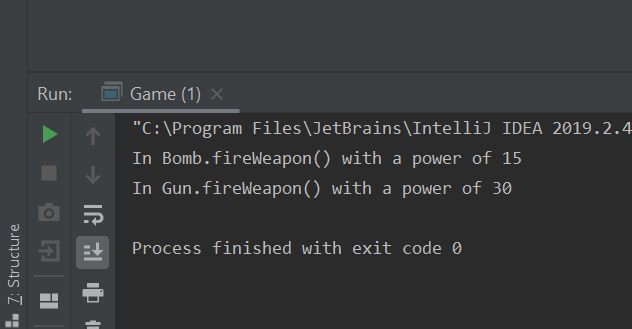
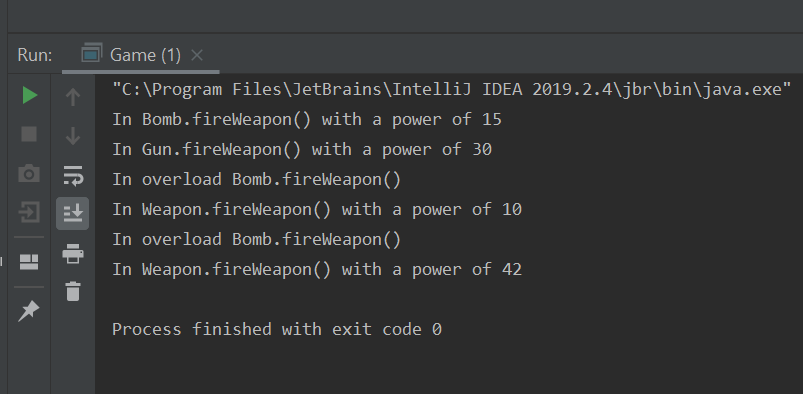
Assignment 5



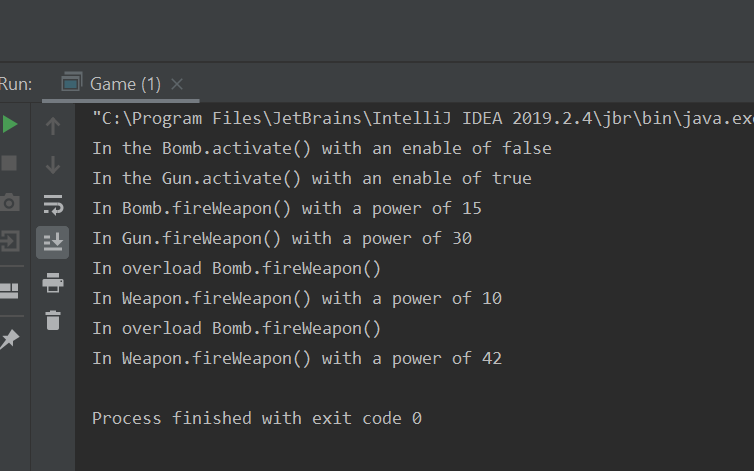
The reason that both bomb and gun are able to use fireWeapon() is because they have inherited the class of Weapon. By using the public Class extends Class, you can pass common methods to other more specialized forms of this class. Since the fireWeapon (or maybe better named as useWeapon for weapons that don’t have triggers) is common amongst all weapons, it can be passed through inheritance. This way you have the ability to type bomb.fireWeapon() as well as gun.fireWeapon() without having to recreate fireWeapon() in each of the child classes.

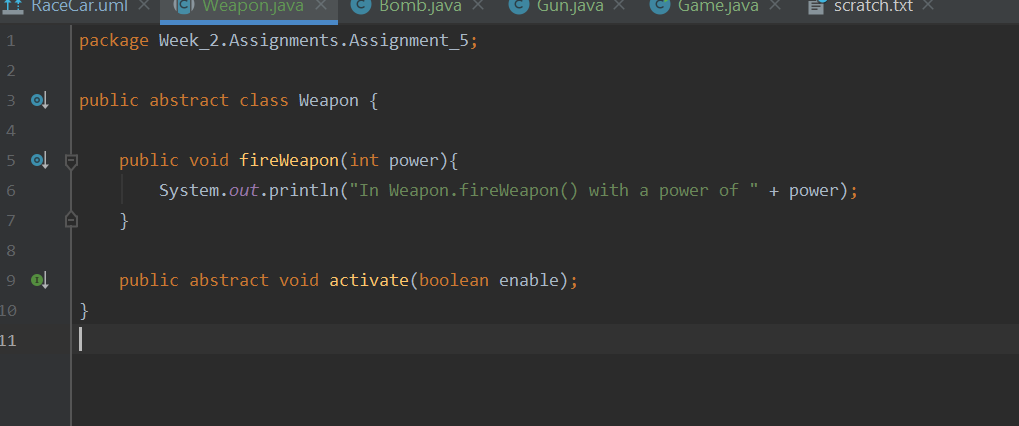


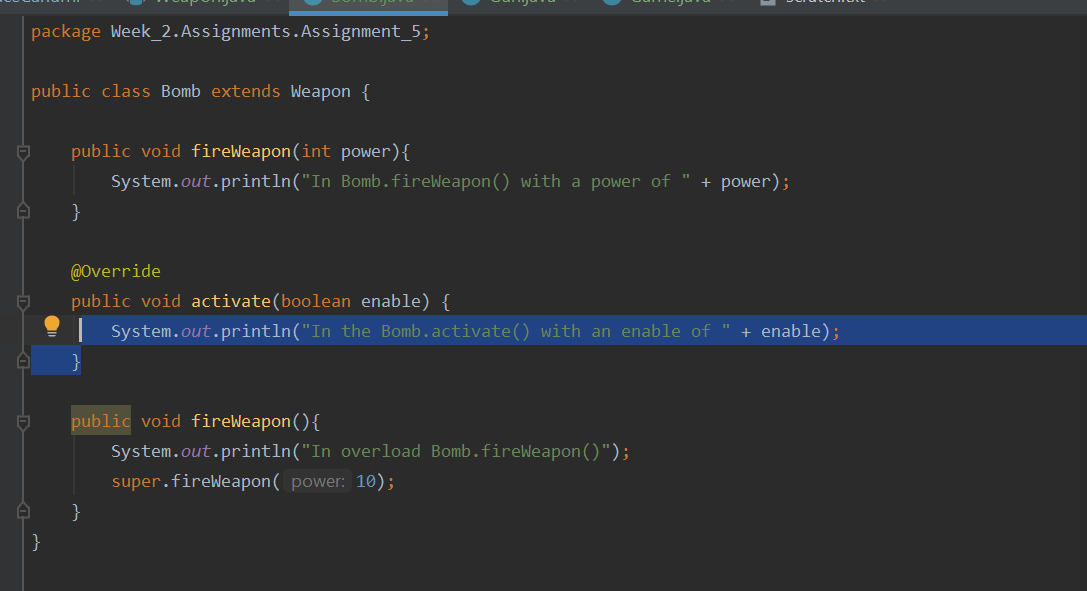
Overriding the method allows you to create a local version of the same method that will display or do something that is specific to that class. It give the used the flexibility to allow for minor customization to a class method to help it meet a need of another class that may be fractionally different.

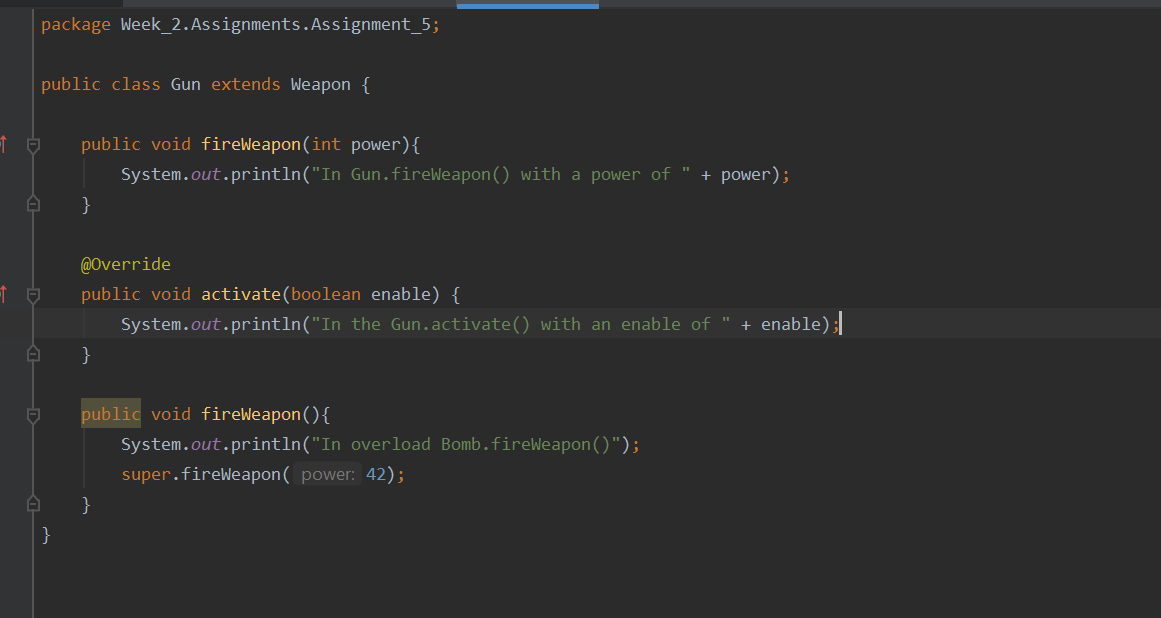


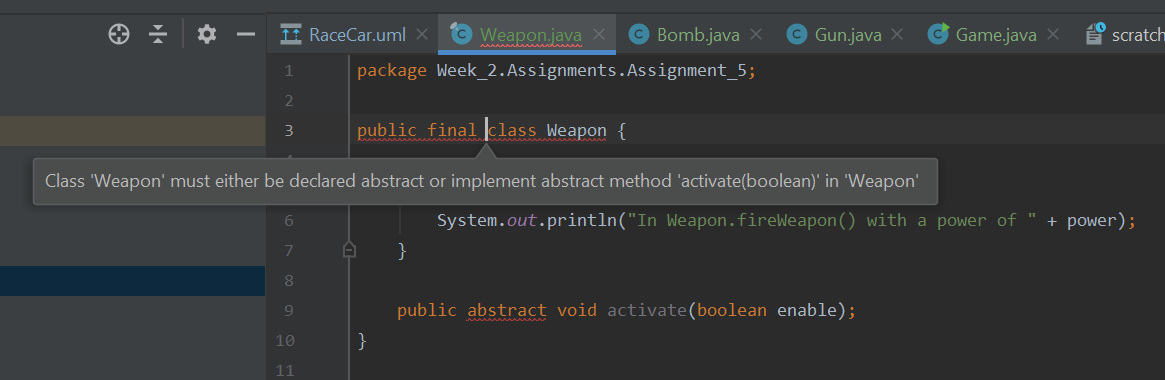
In overloading a method, you may want to have the ability to offer either a String or an int input. In this case the overloaded method just calls its parent to execute. You can manually set a power in case a user doesn’t know what the power should be by default and provide an output. Likewise you could also do some calculations that are specific to the type of weapon that may not affect the parent. You could say that all weapons do a base power of 10 and then have an overloaded method that allows for a calculation of power based on the weapon being used, IE: 10\*n.



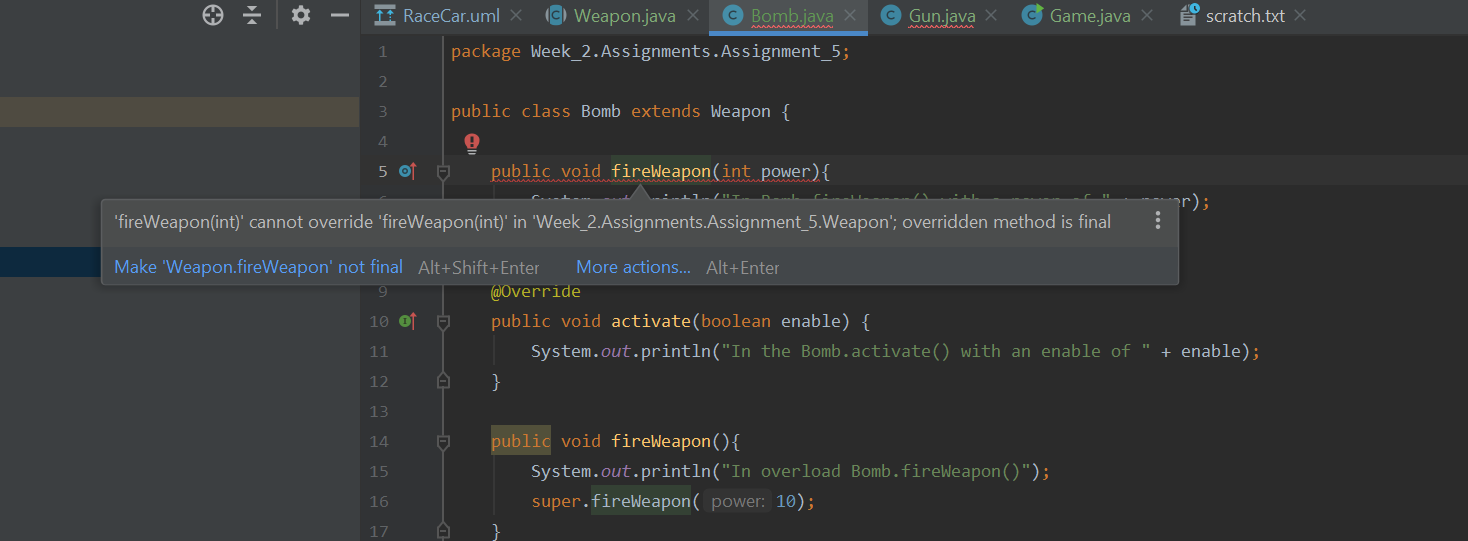








This occurs because both final and abstract are mutually exclusive to each other. Abstract classes require to be overridden where final classes cannot be overridden. Therefore if you make an abstract class final, by its nature you could never use or call it.



The reason for the errors when making Weapon.fireWeapon() final is for the same reason as stated in making the Weapon class final. Final means that the class cannot be overridden in another class that extends it. You can use this to protect certain public classes from being overridden and changing how they should operate.