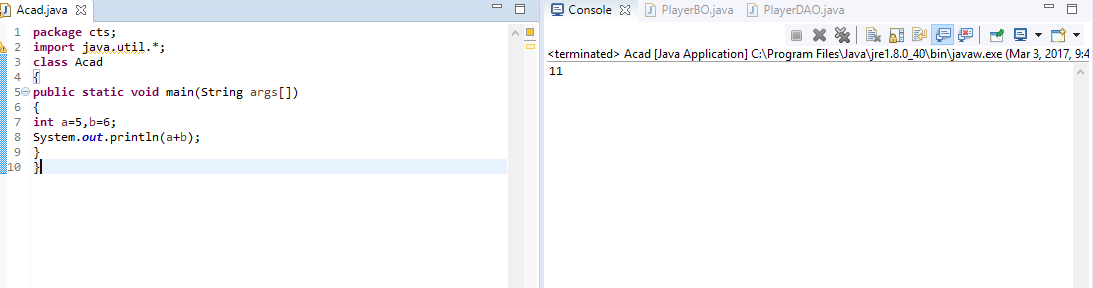
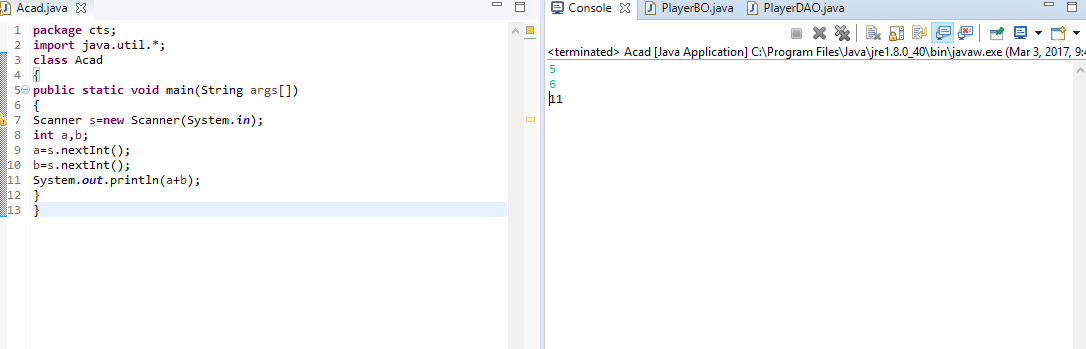
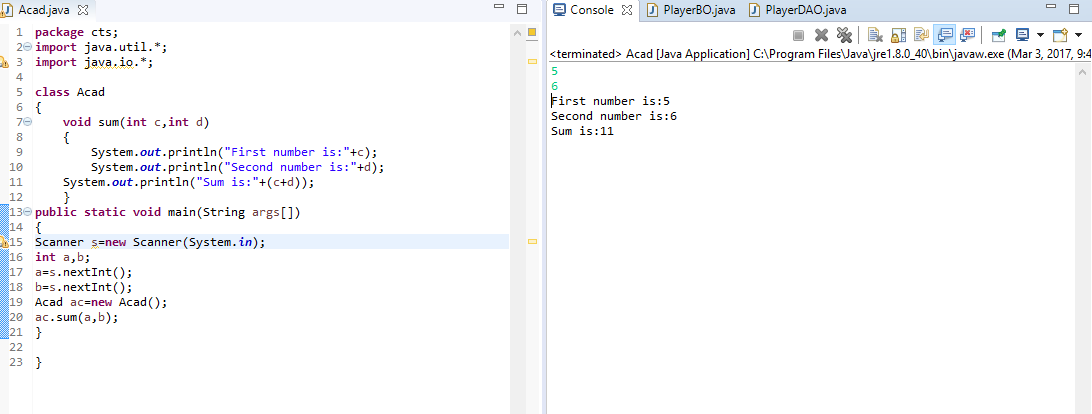
1.



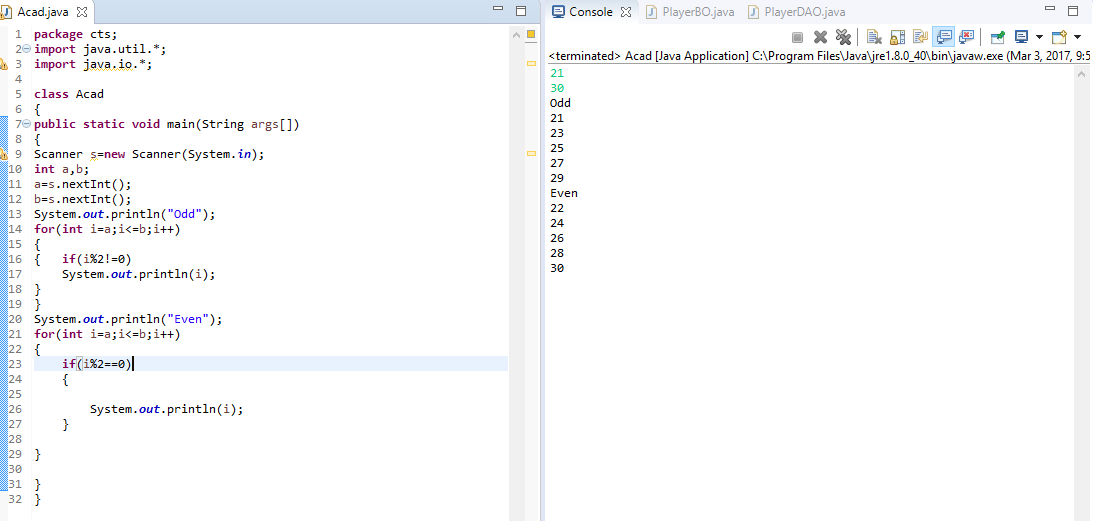
2.



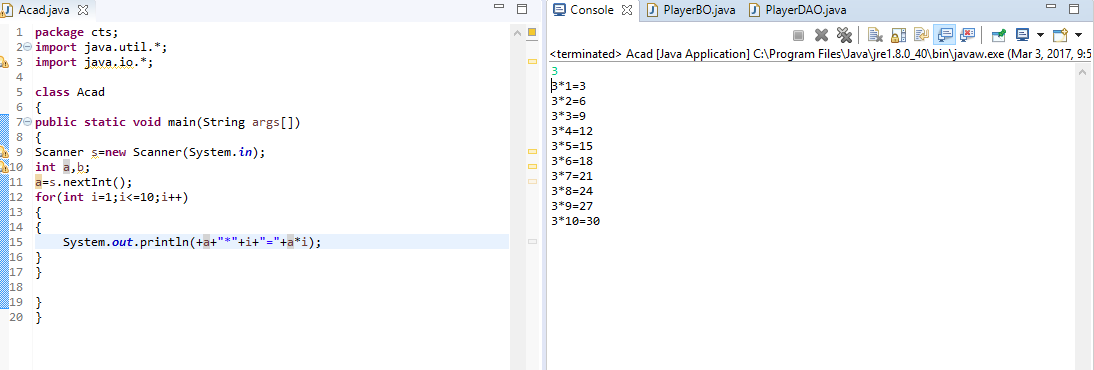
3.



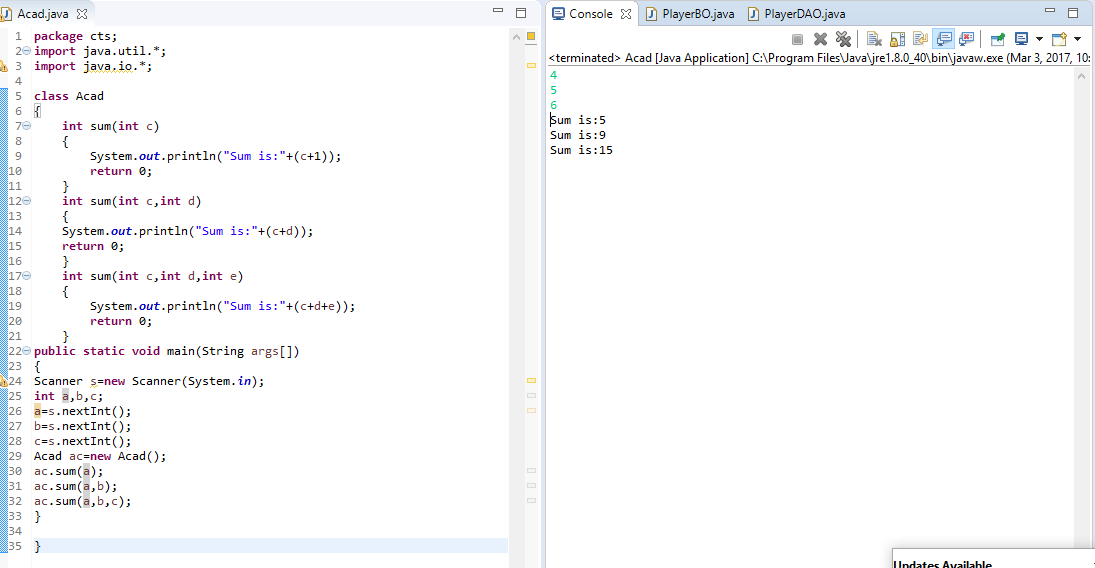
4.



5.



6.



7.

In Java it is possible to define two or more methods within the same class that share the same name, as long as their parameter declarations are different. When this is the case, the methods are said to be overloaded, and the process is referred to as method overloading. Method overloading is one of the ways that Java implements polymorphism.

class OverloadDemo {

void test() {

System.out.println("No parameters");

}

void test(int a) {

System.out.println("a: " + a);

}

void test(int a, int b) {

System.out.println("a and b: " + a + " " + b);

}

double test(double a) {

System.out.println("double a: " + a);

return a\*a;

}

}

class Overload {

public static void main(String args[]) {

OverloadDemo ob = new OverloadDemo();

double result;

ob.test();

ob.test(10);

As you can see, test( ) is overloaded four times. The first version takes no parameters, the second takes one integer parameter, the third takes two integer parameters, and the fourth takes one double parameter. The fact that the fourth version of test( ) also returns a value is of no consequence relative to overloading, since return types do not play a role in overload resolution.

ob.test(10, 20);

result = ob.test(123.2);

System.out.println("Result of ob.test(123.2): " + result);

}

8.

