OOPS SOLUTIONS

Solution 1:

```
import java.util.*;
class Complex{
 int real;
 int imag;
 public Complex(int r, int i){
   real = r;
   imag = i;
 }

 public static Complex add(Complex a, Complex b){
   return new Complex((a.real+b.real),(a.imag+b.imag));
 }

 public static Complex diff(Complex a, Complex b){
   return new Complex((a.real-b.real),(a.imag-b.imag));
 }

 public static Complex product(Complex a, Complex b){
                                                    return                    new
Complex(((a.real*b.real)-(a.imag*b.imag)),((a.real*b.imag)+(a.imag*b.real)));
 }

 public void printComplex(){
   if(real == 0 && imag!=0){
     System.out.println(imag+"i");
   }
   else if(imag == 0 && real!=0){
     System.out.println(real);
   }
   else{
     System.out.println(real+"+"+imag+"i");
   }
 }

}
```

```
class Solution {
 public static void main(String[] args){
   Complex c = new Complex(4,5);
   Complex d = new Complex(9,4);

   Complex e = Complex.add(c,d);
   Complex f = Complex.diff(c,d);
   Complex g = Complex.product(c,d);
   e.printComplex();
   f.printComplex();
   g.printComplex();
 }
}
```

Solution 2: B. Driving electric car

The drive() method in the Car class does not override the version in the Automobile class since the method is not visible to the Car class.

The drive() method in the ElectricCar class is a valid override of the method in the Car class, with the access modifier expanding in the subclass. For these reasons, the code compiles, and Option D is incorrect.

In the main() method, the object created is an ElectricCar, even if it is assigned to a Car reference. Due to polymorphism, the method from the ElectricCar will be invoked, making Option B the correct answer.

Solution 3 : B. public and protected both can be used.

You can provide only a less restrictive or same-access modifier when overriding a method.

Solution 4 : C. 13245

The class is loaded first, with the static initialization block called and 1 is outputted first. When the BlueCar is created in the main() method, the superclass initialization happens first. The instance initialization blocks are executed before the constructor, so 32 is outputted next. Finally, the class is loaded with the instance initialization blocks again being called before the constructor, outputting 45. The result is that 13245 is printed, making Option C the correct answer.