Assignment of interface

1. Because when we don't declare any access specifier to method in class it assigned as Default and in interface by default it assigned as public static
   So **it reduces the visibility of the methodX() from public to default**

```
interface X
{
    void methodX();
}
class Y implements X{
    public void methodX()
    {
        System.out.println(x:"Method X");
    }
}
```

2. it doesn't compile successfully because **default access modifier of variables in interface in public static final**, thus it **cannot be change** in implemented class.

3. **Class cannot be implements to interface** because methods in class can have body with it but method in interface are declared as a abstract methods.

4. **1-QQQQPPPP 2-PPPPQQQQ**

5. Sure, the above program will run correctly.
   The output of the program will be **4**, because return i =+ i * i ;
   Firstly it will multiply i with i that is 4 then =+ assignes the value of before + operator to the variable after = .

6. **What is an interface in Java?**
   Ans:
   Interface is an blueprint of the class. It is an collection of abstract methods and static final variables. We can use interface to achieve abstraction in java.
   We can only show the essential implementation of the methods in interface and declare that methods in implemented class.

7. **Which modifiers are allowed for methods in an Interface?**

   Ans:

   Only public and default access modifiers are allowed in interface.
   When we give private and protected modifiers to the methods interface it gives compilation error.

8. **Suppose A is an interface. Can we create an object using new A()?**

   Ans:

Interface is basically a complete abstract class. That means Interface only have deceleration of method not their implementation. So if we don't have any implementation of a method then that means if we create object of that interface and call that method it compile nothing as there is no code to compile.

To overcome this, firstly we have to create class that implements that interface means that class is implementing the methods of that interface. Now when we create object of this class it gives us the permission to access all methods of interface with their implementation in the class.

9. **Can an interface extends another interface in Java?**

   Ans:

   Yes, interface extends another interface. Its like a inheritance to the interface when we extends one interface with other. We can use the abstract methods of one interface to the extended interface.