Java concepts on Innerclass

Q1) What is an inner class?

Ans) Inner class is a class defined inside other class and act like a member of the enclosing class.

Q2) What are the different types of inner classes?

Ans) There are two main types of inner classes –

- Static member class
- Inner class
  - Member class
  - Anonymous class
  - Local class

Q3) What is static member class?

Ans) A static member class behaves much like an ordinary top-level class, except that it can access the static members of the class that contains it. The static nested class can be accessed as the other static members of the enclosing class without having an instance of the outer class. The static class can contain non-static and static members and methods.

```
public class InnerClass {
    static class StaticInner {
        static int i = 9;
        int no = 6;
        private void method() {}
        public void method1() {}
        static void method2() {}
        final void method3() {}
    }
}
```

The static inner class can be accessed from Outer Class in the following manner:

```
InnerClass.StaticInner staticObj= new InnerClass. StaticInner ();
```

No outer class instance is required to instantiate the nested static class because the static class is a static member of the enclosing class.

Q4) What are non static inner classes?

Ans) The different type of static inner classes are: Non - static inner classes – classes associated with the object of the enclosing class.

Member class - Classes declared outside a function (hence a "member") and not declared "static".
The member class can be declared as public, private, protected, final and abstract. E.g.

```
public class InnerClass {
    class MemberClass {
        public void  method() { }
    }
}
```

Method local class – The inner class declared inside the method is called method local inner class. Method local inner class can only be declared as final or abstract. Method local class can only access global variables or method local variables if declared as final

```
public class InnerClass {
    int i = 9;
    public void method1() {
        final int k = 6;
        class MethodLocal {
          MethodLocal() {
            System.out.println(k  + i);
          }
        }
    }
}
```

Anonymous inner class - These are local classes which are automatically declared and instantiated in the middle of an expression. Also, like local classes, anonymous classes cannot be public, private, protected, or static. They can specify arguments to the constructor of the superclass, but cannot otherwise have a constructor. They can implement only one interface or extend a class.
Anonymous class cannot define any static fields, methods, or classes, except for static final constants.
Also, like local classes, anonymous classes cannot be public, private, protected, or static

Some examples:

```
public class MyFrame extends JFrame {
    JButton btn = new JButton();
    MyFrame() {
        btn.addActionListener(new  ActionListener() {
            public void  actionPerformed(ActionEvent e) {}
        });
    }
}
//Anonymous class used with comparator
List<Parent> l = new ArrayList<Parent>();
l.add(new Parent(2));
l.add(new Parent(3));
Collections.sort(l, new Comparator() {
    public int compare(Object o1,  Object o2) {
        Parent prt1 = (Parent) o1;
        Parent prt2 = (Parent) o2;
        if (prt1.getAge() > prt2.getAge()) {
            return -1;
        }else if(prt1.getAge()<prt2.getAge())  {
            return 1;
        } else {
            return 0;
        }
    }
});
```

Q5)Can a static nested class have access to the enclosing class non-static methods or instance variables?

Ans) No .

Q6)What are the advantages of Inner classes?

Ans) The embedding of inner class into the outer class in the case when the inner class is to be used only by one class i.e. the outer class makes the package more streamlined. Nesting the inner class code where it is used (inside the outer class) makes the code more readable and maintainable

The inner class shares a special relationship with the outer class i.e. the inner class has access to all members of the outer class and still have its own type is the main advantages of Inner class. Advantage of inner class is that they can be hidden from the other classes in the same package and still have the access to all the members (private also) of the enclosing class. So the outer class members which are going to be used by the inner class can be made private and the inner class members can be hidden from the classes in the same package. This increases the level of encapsulation.

If a class A is written requires another class B for its own use, there are two ways to do this. One way is to write a separate class B or to write an inner class B inside class A. Advantage of writing the inner class B in the class A is you can avoid having a separate class. Inner classes are best used in the event handling mechanism and to implement the helper classes. The advantage of using inner class for event handling mechanism is that the use of if/else to select the component to be handled can be avoided. If inner classes are used each component gets its own event handler and each event handler implicitly knows the component it is working for.

Q7)What are disadvantages of using inner classes?

Ans)

- Using inner class increases the total number of classes being used by the application. For all the classes created by JVM and loaded in the memory, jvm has to perform some tasks like creating the object of type class. Jvm may have to perform some routine tasks for these extra classes created which may result slower performance if the application is using more number of inner classes.
- Inner classes get limited support of ide/tools as compared to the top level classes, so working with the inner classes is sometimes annoying for the developer.

Q8) What are different types of anonymous classes?

Ans 1) **Plain old anonymous class type one**–

```
class superClass{
    void doSomething() {
        System.out.println("Doing something in the Super class");
    }
}
class hasAnonymous{
    superClass anon = new superClass(){
    void doSomething() {
        System.out.println("Doing something in the Anonymous class");
    }
};
```

Here anon is the reference which is of type superClass which is the class extended by the anonymous class i.e. superclass of the anonymous class. The method doSomething() is the super class method overridden by the anonymous class.

Plain old anonymous class type two –

```
interface Eatable {
  public void prepareSweets();
}
class serveMeal {
    Eatable food = new Eatable(){
    public void
        prepareSweets(){ //come implementation code goes here }
    };
}
```

food is reference variable of type Eatable interface which refers to the anonymous class which is the implementer of the interface Eatable. The anonymous implementer class of the interface Eatable implements its method prepareSweets() inside it.

Argument defined anonymous class –

```
interface Vehicle {
    void getNoOfWheels();
}
class Car {
    void getType(Vehical v) { }
}
class BeautifulCars {
    void getTheBeautifilCar() {
        Car c = new Car ();
        c.getType (new Vehicle () {
            public void getNoOfWheels () {
                System.out.println("It has four wheels");
            }
        });
    }
}
```

Anonymous class is defined as the argument of the method getTheBeautifilCar(), this anonymous class is the implementer of the interface Vehicle. The method of class Car getTheBeautifilCar() expects the argument as an object of type Vehicle. So first we create an object of Car referenced by the variable 'c'. On this object of Car we call the method getTheBeautifilCar() and in the argument we create an anonymous class in place which is the implementer of interface Vehicle hence of type Vehicle.

Q9) If you compile a file containing inner class how many .class files are created and what are all of them accessible in usual way?

Ans) If a inner class enclosed with an outer class is compiled then one .class file for each inner class an a .class file for the outer class is created. e.g.

```
class EnclosingOuter {
    class Inner{ }
}
```

If you compile the above code with command

% javac EnclosingOuter.java
Two files are created will be created. Though a separate inner class file is generated, the inner class file is not accessible in the usual way.

EnclosingOuter.class
EnclosingOuter$Inner.class

Q10) How to access the inner class from code within the outer class?

Ans) The inner class is instantiated only through the outer class instance.

```
class EnclosingOuter {
    private int noInnerClass = 1;
    public void getNoOfInnerClasses(){
        Inner in = new Inner();
        System.out.println("No Of Inner classes is :"+ in.getNoOfClassesFromOuter());
    }
class Inner{
    public int getNoOfClassesFromOuter(){
        return noInnerClass;
    }
}
```

Here the method getNoOfInnerClasses() is called on the outer class's instance through this outer class instance the inner class instance in is created.

Recommend Reading

- How java manages Memory?
- Use of hashcode and equals
- Comparable and Comparator in java
- How to create Singleton class in Java?
- Difference between equals and ==?
- Data structures
- Cap Theorem in Details
- Why is it recommended to use Immutable objects in Java
- Complete list of Java articles

Recommend Reading
- How java manages Memory?
- Why is it advised to use hashcode and equals together?
- Comparable and Comparator in java
- How to create Singleton class in Java?
- Difference between equals and ==?
- When to use abstract class over interface?
- Difference between final, finally and finalize
- Why is it recommended to use Immutable objects in Java