Template Method Design Pattern

Template method design pattern is a behavioral design pattern of the gang of four design pattern. Template method defines an algorithm in which an abstract base class contains an template method that contains sequence of method execution in an order.

Requirement: - Assume we have a series of functionality to be invoked in a particular order and few functionality are common where few functionalities are implemented dependent.

Problem: - Remembering multiple method names and their sequence of invocation that are required to complete certain task is complex and error prone process.

Solution: - Work with Template Method design pattern, where one java method definition contains all the method calls in a sequence. So programmer needs to call only one java method to complete the task.

Steps for implementing Template Method Design Pattern

1. Define one abstract class containing abstract method and concrete method. The concrete method are the methods with common logics whereas abstract methods are the methods whose logic is depends on the implementation class and it will be defined in the child classes.

Coding:-

Problem:-

HireFresher.java

```java
package com.nt.hire;

public abstract class HireFresher {

	public boolean conductApptitudeTest() {
		System.out.println("conducting Apptitude Test");
		return true;
	}

	public boolean conductGD() {
		System.out.println("Conducting Grop Disscussion");
		return true;
	}

	public boolean conductHR() {
		System.out.println("Condcuting HR inteviwew");
		return true;
	}

	public abstract boolean conductTechnicalTest();

	public abstract boolean conductSystemTest();

}
```

JavaHireFresher.java

```java
package com.nt.hire;

public class JavaHireFresher extends HireFresher {

	@Override
	public boolean conductTechnicalTest() {
		System.out.println("Java conduct Technical Test");
		return true;
	}

	@Override
	public boolean conductSystemTest() {
		System.out.println("Java conduct System Test");
		return true;
	}

}
```

DotNetHireFresher.java

```
package com.nt.hire;

public class DotNetHireFresher extends HireFresher {

	@Override
	public boolean conductTechnicalTest() {
		System.out.println("DotNet conducting Technical Test");
		return true;
	}

	@Override
	public boolean conductSystemTest() {
		System.out.println("Dotnet conducting system test");
		return true;
	}

}
```

RecurimentTest1.java

```
	package com.nt.test;

	import com.nt.hire.HireFresher;
	import com.nt.hire.JavaHireFresher;

	public class RecurimentTest1 {

		public static void main(String[] args) {
			HireFresher fresher=null;
			boolean result=false;
			//create the object
			fresher=new JavaHireFresher();
			result=fresher.conductApptitudeTest();
			if(result)
				result=fresher.conductGD();
			if(result)
				result=fresher.conductTechnicalTest();
			if(result)
				result=fresher.conductSystemTest();
			if(result)
				result=fresher.conductHR();

			if(result) {
				System.out.println("U   are   selected   for   java
      developer");
                  }
                  else {
                        System.out.println("U are not selected for java
      developer");
                  }
            }//main

}//class
```

RecurimentTest2.java

```
package com.nt.test;

import com.nt.hire.DotNetHireFresher;
import com.nt.hire.HireFresher;
import com.nt.hire.JavaHireFresher;

public class RecurimentTest2 {

     public static void main(String[] args) {
          HireFresher fresher = null;
          boolean result = false;
          // create the object
          fresher = new DotNetHireFresher();
          result = fresher.conductApptitudeTest();
          if (result)
               result = fresher.conductGD();
          if (result)
               result = fresher.conductTechnicalTest();
          if (result)
               result = fresher.conductSystemTest();
          if (result)
               result = fresher.conductHR();

          if (result) {
               System.out.println("U  are  selected  for  Dotnet
developer");
          } else {
               System.out.println("U are not selected for DotNet
developer");
          }
     }// main
}// class
```

Solutiuons:-

HireFresher.java

```
package com.nt.hire;

public abstract class HireFresher {

	public boolean conductApptitudeTest() {
		System.out.println("conducting Apptitude Test");
		return true;
	}

	public boolean conductGD() {
		System.out.println("Conducting Grop Disscussion");
		return true;
	}

	public boolean conductHR() {
		System.out.println("Condcuting HR inteviwew");
		return true;
	}

	public abstract boolean conductTechnicalTest();

	public abstract boolean conductSystemTest();

	public boolean recurimentProcess() { // Template Method
		boolean result = false;
		result = conductApptitudeTest();
		if (result)
			result = conductGD();
		if (result)
			result = conductTechnicalTest();
		if (result)
			result = conductSystemTest();
		if (result)
			result = conductHR();
		return result;
	}

}
```

JavaHireFresher.java

```java
package com.nt.hire;

public class JavaHireFresher extends HireFresher {

	@Override
	public boolean conductTechnicalTest() {
		System.out.println("Java conduct Technical Test");
		return true;
	}

	@Override
	public boolean conductSystemTest() {
		System.out.println("Java conduct System Test");
		return true;
	}

}
```

DotNetHireFresher.java

```java
package com.nt.hire;

public class DotNetHireFresher extends HireFresher {

	@Override
	public boolean conductTechnicalTest() {
		System.out.println("DotNet conducting Technical Test");
		return true;
	}

	@Override
	public boolean conductSystemTest() {
		System.out.println("Dotnet conducting system test");
		return true;
	}

}
```

HireFresherFactory.java

```
	package com.nt.factory;

	import com.nt.hire.DotNetHireFresher;
	import com.nt.hire.HireFresher;
	import com.nt.hire.JavaHireFresher;

	public class HireFresherFactory {

		public static HireFresher getInstance(String domain) {
			HireFresher fresher = null;
			if (domain.equalsIgnoreCase("java")) {
				fresher = new JavaHireFresher();
			} else if (domain.equalsIgnoreCase("dotnet")) {
				fresher = new DotNetHireFresher();
			} else {
				throw new IllegalArgumentException("Invalid Domain
	name");
			}
			return fresher;
		}
}
```

RecurimentTest1.java

```
	package com.nt.test;

	import com.nt.factory.HireFresherFactory;
	import com.nt.hire.HireFresher;
	import com.nt.hire.JavaHireFresher;

	public class RecurimentTest1 {

		public static void main(String[] args) {
			HireFresher fresher = HireFresherFactory.getInstance("java");
			boolean result = fresher.recurimentProcess();
```

```
                if (result) {
                        System.out.println("U are selected for java developer");
                } else {
                        System.out.println("U are not selected for java developer");
                }
        }// main

}// class
```

RecurimentTest2.java

```
      package com.nt.test;

      import com.nt.factory.HireFresherFactory;
      import com.nt.hire.DotNetHireFresher;
      import com.nt.hire.HireFresher;
      import com.nt.hire.JavaHireFresher;

      public class RecurimentTest2 {

            public static void main(String[] args) {
                  HireFresher fresher =
      HireFresherFactory.getInstance("dotnet");
                  boolean result = fresher.recurimentProcess();
                  if (result) {
                        System.out.println("U are selected for Dotnet
      developer");
                  } else {
                        System.out.println("U are not selected for DotNet
      developer");
                  }
            }// main

}// class
```