ABSTRACT FACTORY DESIGN PATTERN

(1) Profession interface :

```
package test.app;

public interface Profession {
void print();
}
```

(2) Doctor class :

```
package test.app;

public class Doctor implements Profession{

	@Override
	public void print() {
		System.out.println("i am a doctor");

	}

}
```

(3) Enginner class :

```
package test.app;

public class Engineer implements Profession {

	@Override
	public void print() {
		System.out.println("i am an engineer");

	}

}
```

(4) Teacher Class :

```
package test.app;

public class Teacher implements Profession{

	@Override
	public void print() {
		System.out.println("i am a teacher");

	}

}
```

(5) TraineeDoctor class :

```
package test.app;

public class TraineeDoctor implements Profession{

	@Override
	public void print() {
		System.out.println("i am a trainne doctor");

	}

}
```

(6) TraineeEngineer class :

```
package test.app;

public class TraineeEngineer implements Profession{

	@Override
	public void print() {
	System.out.println("i am trainee engineer");

	}

}
```

(7) TraineeTeacher class :

```
package test.app;

public class TraineeTeacher implements Profession{

	@Override
	public void print() {
	System.out.println("i am a trainee teacher");

	}

}
```

(8) AbstractFactory abstract class :

```
package test.app;

public abstract class AbstractFactory {
public abstract Profession getProfession(String typeOfProfession);
}
```

(9) ProfessionAbstractFactory class :

```java
package test.app;

public class ProfessionAbstractFactory extends AbstractFactory{

	@Override
	public Profession getProfession(String typeOfProfession) {
		if(typeOfProfession==null) {
			return null;
		}else if (typeOfProfession.equalsIgnoreCase("Doctor")) {
			return new Doctor();
		}else if (typeOfProfession.equalsIgnoreCase("Engineer")) {
			return new Engineer();
		}else if (typeOfProfession.equalsIgnoreCase("Teacher")) {
			return new Teacher();
		}
		return null;
	}

}
```

(10) TraineeProfessionAbstractFactory class :

```java
package test.app;
public class TraineeProfessionAbstractFactory extends AbstractFactory{

	@Override
	public Profession getProfession(String typeOfProfession) {
		if(typeOfProfession==null) {
			return null;
		}else if (typeOfProfession.equalsIgnoreCase("Doctor")) {
			return new TraineeDoctor();
		}else if (typeOfProfession.equalsIgnoreCase("Engineer")) {
			return new TraineeEngineer();
		}else if (typeOfProfession.equalsIgnoreCase("Teacher")) {
			return new TraineeTeacher();
		}
		return null;
	}

}
```

(11) AbstractFactoryProducer class :

```java
package test.app;

public class AbstractFactoryProducer {
public static AbstractFactory getProfession(boolean isTrainee) {
	if(isTrainee) {
		return new TraineeProfessionAbstractFactory();
	}else {
		return new ProfessionAbstractFactory();
	}
}
}
```

(12) Main class :

```
package test.app;

public class Main {
public static void main(String [] args) {
	AbstractFactory abstractFactory = AbstractFactoryProducer.getProfession(true);
	Profession profession = abstractFactory.getProfession("Doctor");
	profession.print();

}
}
```