A Factory Pattern or Factory Method Pattern says that just **define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate.** In other words, subclasses are responsible to create the instance of the class.

The Factory Method Pattern is also known as **Virtual Constructor.**

Advantage of Factory Design Pattern

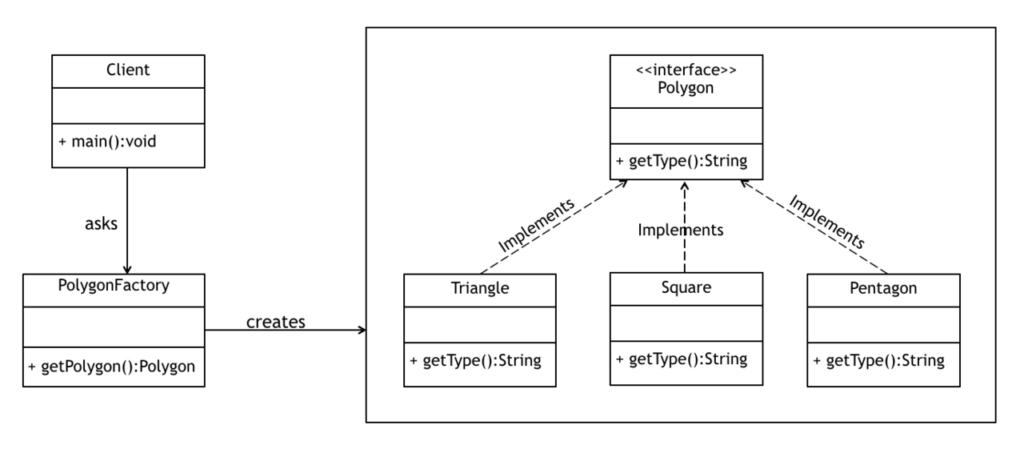
* Factory Method Pattern allows the sub-classes to choose the type of objects to create.
* It promotes the **loose-coupling** by eliminating the need to bind application-specific classes into the code. That means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class.

### Factory Design Pattern Examples in JDK

1. java.util.Calendar, ResourceBundle and NumberFormat getInstance() methods uses Factory pattern.
2. valueOf() method in wrapper classes like Boolean, Integer etc.

### **. When to Use Factory Method Design Pattern**

* When the implementation of an interface or an abstract class is expected to change frequently
* When the current implementation cannot comfortably accommodate new change
* When the initialization process is relatively simple, and the constructor only requires a handful of parameters



public interface Polygon {

String getType();

}

public class Triangle implements Polygon {

@Override

public String getType() {

return "Triangle";

}

}

public class Square implements Polygon {

@Override

public String getType() {

return "Square";

}

}

public class Pentagon implements Polygon {

@Override

public String getType() {

return "Pentagon";

}

}

public class Octagon implements Polygon {

@Override

public String getType() {

return "Octagon";

}

}

public class Heptagon implements Polygon {

@Override

public String getType() {

return "Heptagon";

}

}

public class FactoryDriver {

public static void main(String[] args) {

Polygon p;

PolygonFactory factory = new PolygonFactory();

//get the shape which has 4 sides

p = factory.getPolygon(4);

System.out.println("The shape with 4 sides is a " + p.getType());

//get the shape which has 4 sides

p = factory.getPolygon(8);

System.out.println("The shape with 8 sides is a " + p.getType());

}

}

public class PolygonFactory {

public Polygon getPolygon(int numberOfSides) {

if(numberOfSides == 3) {

return new Triangle();

}

if(numberOfSides == 4) {

return new Square();

}

if(numberOfSides == 5) {

return new Pentagon();

}

if(numberOfSides == 7) {

return new Heptagon();

}

else if(numberOfSides == 8) {

return new Octagon();

}

return null;

}

}