Modèles de conception - Modèle de prototype

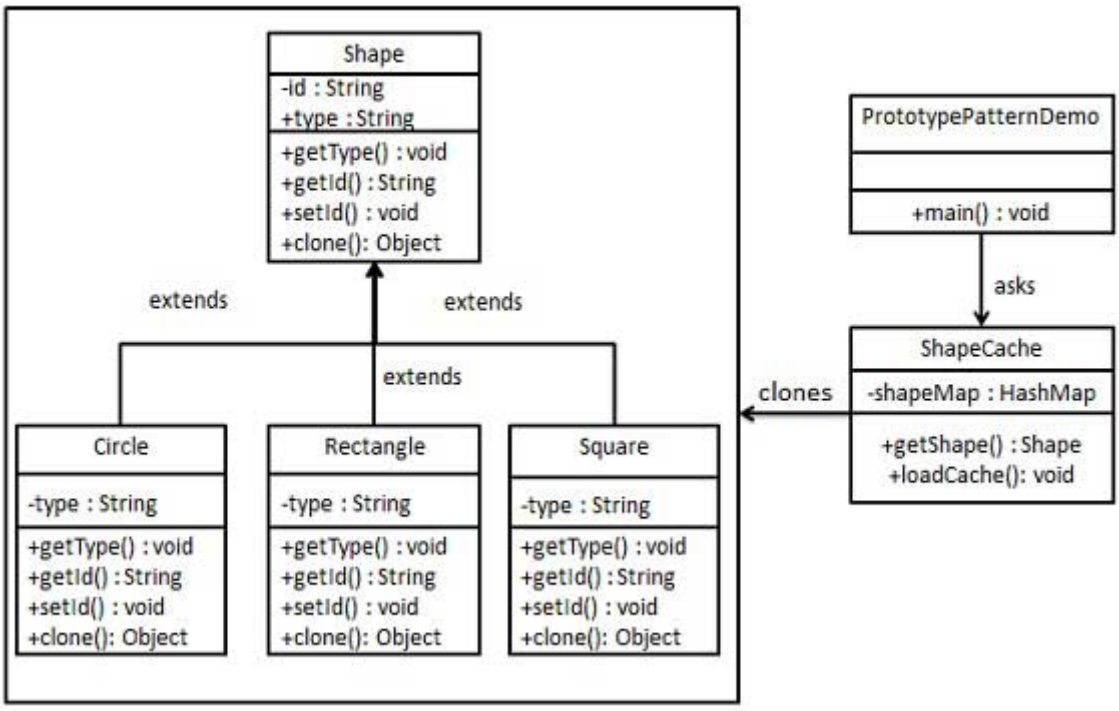
Le modèle de prototype fait référence à la création d'objets en double tout en gardant à l'esprit les performances. Ce type de modèle de conception relève du modèle de création, car ce modèle offre l'une des meilleures façons de créer un objet.

Ce modèle implique la mise en œuvre d'une interface prototype qui indique de créer un clone de l'objet actuel. Ce modèle est utilisé lorsque la création d'un objet directement est coûteuse. Par exemple, un objet doit être créé après une opération de base de données coûteuse. Nous pouvons mettre en cache l'objet, retourner son clone à la prochaine demande et mettre à jour la base de données au fur et à mesure, réduisant ainsi les appels à la base de données.

la mise en oeuvre

Nous allons créer une classe abstraite *Shape* et des classes concrètes étendant la classe *Shape* . Une classe *ShapeCache* est définie comme une forme étape suivante qui stocke des objets dans un *Hashtable* et retourne leur clone sur demande.

PrototypPatternDemo , notre classe de démonstration utilisera la classe *ShapeCache* pour obtenir un objet *Shape* .



Étape 1

Créez une classe abstraite implémentant *une* interface *Clonable* .

Shape.java

```
public abstract class Shape implements Cloneable {
```

```
   private String id;
   protected String type;

   abstract void draw();

   public String getType(){
      return type;
   }

   public String getId() {
      return id;
   }

   public void setId(String id) {
      this.id = id;
   }

   public Object clone() {
      Object clone = null;

      try {
         clone = super.clone();

      } catch (CloneNotSupportedException e) {
         e.printStackTrace();
      }

      return clone;
   }
}
```

Étape 2

Créez des classes concrètes étendant la classe ci-dessus.

Rectangle.java

```
public class Rectangle extends Shape {

   public Rectangle(){
     type = "Rectangle";
   }

   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

Square.java

```
public class Square extends Shape {

   public Square(){
```

```
      type = "Square";
   }

   @Override
   public void draw() {
      System.out.println("Inside Square::draw() method.");
   }
}
```

Circle.java

```
public class Circle extends Shape {

   public Circle(){
     type = "Circle";
   }

   @Override
   public void draw() {
      System.out.println("Inside Circle::draw() method.");
   }
}
```

Étape 3

Créez une classe pour obtenir des classes concrètes de la base de données et stockez-les dans une *table de hachage* .

ShapeCache.java

```
import java.util.Hashtable;

public class ShapeCache {

   private static Hashtable<String, Shape> shapeMap  = new Hashtable<String, Shape>();

   public static Shape getShape(String shapeId) {
      Shape cachedShape = shapeMap.get(shapeId);
      return (Shape) cachedShape.clone();
   }

   // for each shape run database query and create shape
   // shapeMap.put(shapeKey, shape);
   // for example, we are adding three shapes

   public static void loadCache() {
      Circle circle = new Circle();
      circle.setId("1");
      shapeMap.put(circle.getId(),circle);

      Square square = new Square();
      square.setId("2");
      shapeMap.put(square.getId(),square);
```

```
      Rectangle rectangle = new Rectangle();
      rectangle.setId("3");
      shapeMap.put(rectangle.getId(), rectangle);
   }
}
```

Étape 4

PrototypePatternDemo utilise *ShapeCache* classe pour obtenir des clones de formes stockées dans un *Hashtable* .

PrototypePatternDemo.java

```
public class PrototypePatternDemo {
   public static void main(String[] args) {
      ShapeCache.loadCache();

      Shape clonedShape = (Shape) ShapeCache.getShape("1");
      System.out.println("Shape : " + clonedShape.getType());

      Shape clonedShape2 = (Shape) ShapeCache.getShape("2");
      System.out.println("Shape : " + clonedShape2.getType());

      Shape clonedShape3 = (Shape) ShapeCache.getShape("3");
      System.out.println("Shape : " + clonedShape3.getType());
   }
}
```

Étape 5

Vérifiez la sortie.

```
Shape : Circle
Shape : Square
Shape : Rectangle
```