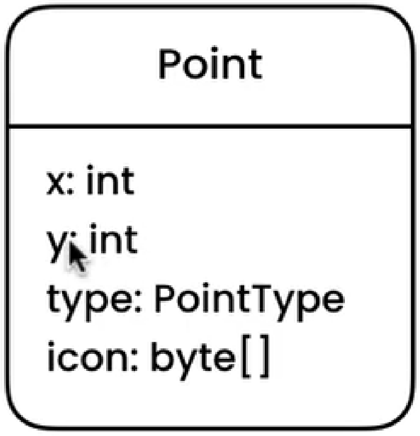
Flyweight Pattern

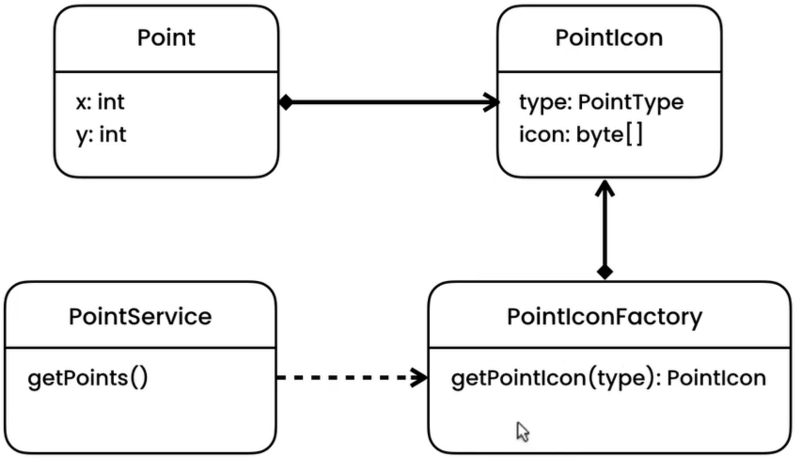
A Flyweight is an object we can share in different contexts.

Useful when dealing with a large number of objects to help manage memory. For instance google maps has to render points on the map - if you store each point image as a byte array taht is around 20kb in size, if you then try to plot 1000 of these that is 20mb which some mobile devices may not have free causing the app to crash. the flyweight pattern helps to solve this problem.

Our point class :



becomes :



We seperate out the type and icon attributes as they are relate e.g. if the type is a cafe then we have a spefic icon for that. We then utilse the PointIcon Factory to compose point icons and store them in memory / cache so that one icon can be share between many Points with the help of the getPoints method.


```
public class Point {
  private int x; // 4 bytes
 private int y; // 4 bytes
 private PointIcon icon;

 public Point(int x, int y, PointIcon icon) {
    this.x = x;
 this.y = y;
 this.icon = icon;
 }

  public void draw() {
    System.out.printf("%s at (%d, %d)", icon.getType(), x, y);
 }
}
```

```
public class PointIcon {
  private final PointType type; // 4 bytes
 private final byte[] icon; // 20 KB -> 20 MB

 public PointIcon(PointType type, byte[] icon) {
    this.type = type;
 this.icon = icon;
 }

  public PointType getType() {
    return type;
 }
}
```

```
public class PointIconFactory {
  private Map<PointType, PointIcon> icons = new HashMap<>();
//ensures that the same icon is not created multiple times in memory

 public PointIcon getPointIcon(PointType type) {
    if (!icons.containsKey(type)) {
      var icon = new PointIcon(type, null);
 icons.put(type, icon);
 }

    return icons.get(type);
 }
}
```

```
public class PointService {
  private PointIconFactory iconFactory;

 public PointService(PointIconFactory iconFactory) {
    this.iconFactory = iconFactory;
 }

  public List<Point> getPoints() {
    List<Point> points = new ArrayList<>();
 var point = new Point(1, 2, iconFactory.getPointIcon(PointType.CAFE));
 points.add(point);

 return points;
 }
}
```

```
public enum PointType {
  HOSPITAL,
 CAFE,
 RESTAURANT
}
```

```
public class Main {
    public static void main(String[] args) {
    var service = new PointService(new PointIconFactory())
    for (var point: servive.getPoints())

point.draw()

}

}
```