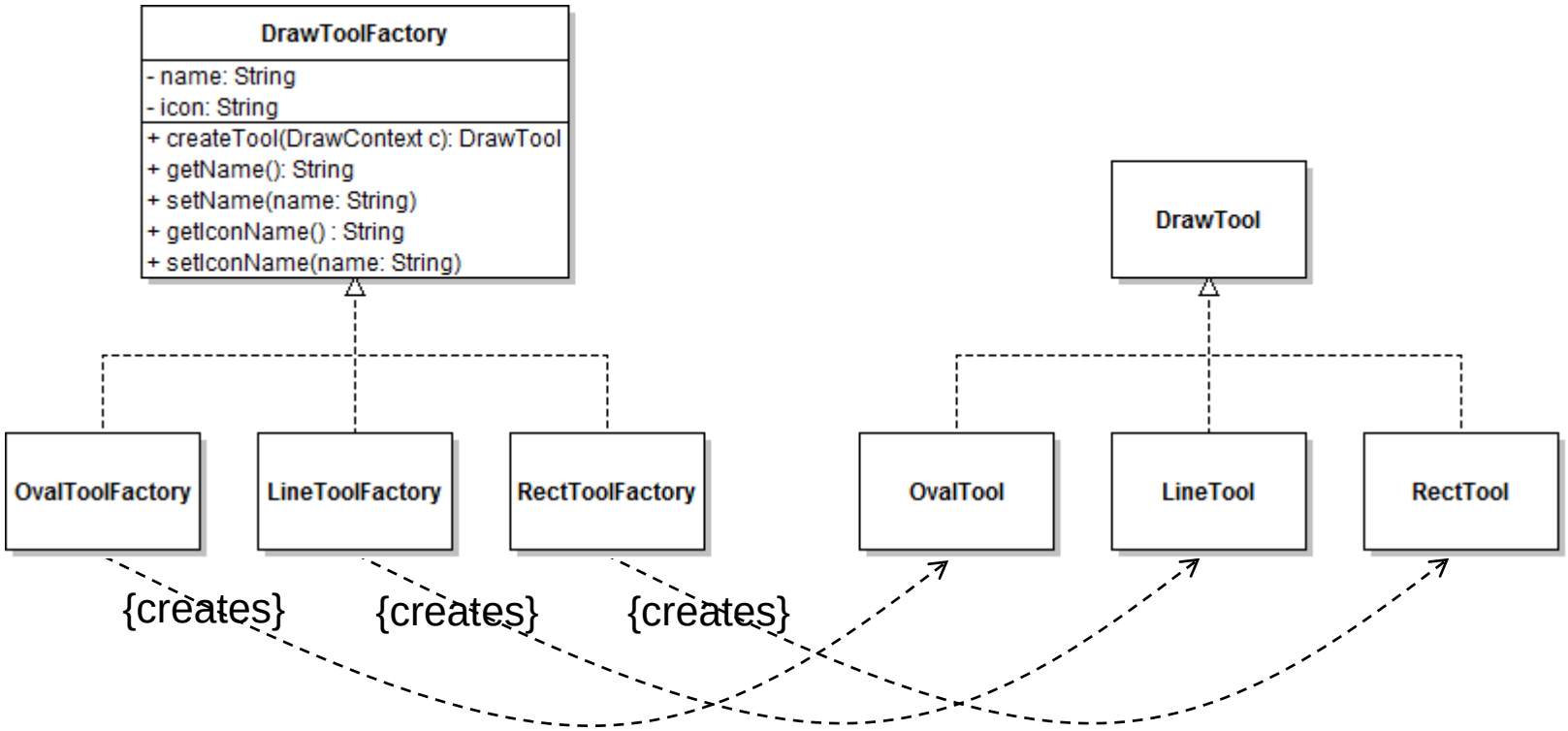
Assignment 11: Configuration



jdraw-context.xml

Tool factories are defined as singletons

```
<bean id="line" class="jdraw.figures.LineToolFactory">
   <property name="name"><value>Line</value></property>
   <property name="iconName"><value>line.png</value></property>
</bean>

<bean id="rect" class="jdraw.figures.RectToolFactory">
   <property name="name"><value>Rectangle</value></property>
   <property name="iconName"><value>rectangle.png</value></property>
</bean>

<bean id="oval" class="jdraw.figures.OvalToolFactory">
   <property name="name"><value>Oval</value></property>
   <property name="iconName"><value>oval.png</value></property>
</bean>
```

Name of the tool and the icon name can be defined in the spring context

ToolFactories

- **OvalToolFactory**

```
public class OvalToolFactory extends AbstractDrawToolFactory {
   @Override
   public DrawTool createTool(DrawContext c) {
      return new OvalTool(c, getName(), getIconName());
   }
}
```

- **LineToolFactory**

```
public class LineToolFactory extends AbstractDrawToolFactory {
   @Override
   public DrawTool createTool(DrawContext c) {
      return new LineTool(c, getName(), getIconName());
   }
}
```

AbstactFactory.java

```
public abstract class AbstractDrawToolFactory
                                          implements DrawToolFactory {
   private String name; // name of the tool
   private String icon; // name of the icon

   @Override
   public void setName(String name) { this.name = name; }

   @Override
   public String getName() { return name; }

   @Override
   public void setIconName(String name) { this.icon = name; }

   @Override
   public String getIconName() { return icon; }

}
```

Invoked by spring

Invoked by spring

jdraw-context.xml

```
<bean id="drawContext" class="jdraw.std.StdContext" scope="prototype"
   init-method="initGUI"
>
   <constructor-arg ref="drawView"/>

   <property name="width"><value>600</value></property>
   <property name="height"><value>400</value></property>

   <constructor-arg>
      <list>
         <ref bean="line"/>
         <ref bean="rectangle"/>
         <ref bean="oval"/>
         <null/>
         <ref bean="ernst-swiss"/>
      </list>
   </constructor-arg>
</bean>
```

Separator in the figure menu

StdContext.java

- **Old version**

```
@Override
protected void doRegisterDrawTools() {
   DrawTool rectangleTool = new RectTool(this);
   addTool(rectangleTool);
   DrawTool ovalTool = new OvalTool(this);
   addTool(ovalTool);
   DrawTool lineTool = new LineTool(this);
   addTool(lineTool);
}
```

StdContext.java

- **New Version**

```
   @Override
   protected void doRegisterDrawTools() {
      for (DrawToolFactory dt : getToolFactories()) {
         addTool(dt == null ? null : dt.createTool(this));
      }
   }
```

```
private List<DrawToolFactory> toolFactories = new LinkedList<>();

public StdContext(DrawView v, List<DrawToolFactory> toolFactories) {
   …; this.toolFactories = toolFactories; …
}

protected final List<DrawToolFactory> getToolFactories() {
   return toolFactories;
}
```

Remarks

- **DrawToolFactories**
  - Used to create DrawTool instances in the initialization for *each* context
  - Draw tools could also be injected directly
    - They would have to be defined as non-singleton beans (scope="prototype"), as they are stateful, i.e. they contain a reference to the draw context in which they are used (e.g. to set the status text)
    - Context references in the tools would have to be set in the Java code

- **Compatibility**
  - Other figures can be included without changing any line of Java code
    - Only works if the interfaces used by the figure implementations have not been changed (`Figure` / `FigureHandle` / `DrawTool`)
      - Not moved into other packages
      - No additional methods (except default methods)