In this laboratory session you will write definitions of three classes: **Vehicle**, **Sedan**, and **Truck**. The **Vehicle** class will be abstract. The **Sedan** and **Truck** classes will extend the **Vehicle** class. They will not be abstract. Once you have written these class definitions, you will be able to test them using a program called **VehiclesApplication**. The following statements appear in the run method of this program:

```java
Vehicle sedan = new Sedan(150,100,300,125);
add(sedan);

Vehicle truck = new Truck(150,300,300,125);
add(truck);
```

These statements cause Java to open a window and draw the pictures shown above. Notice that the body and the wheels of the sedan and truck are the same. On the other hand, the canopy of the sedan differs from the canopy of the truck. This observation leads to a strategy for designing and implementing the three classes. The **Vehicle** class should be responsible for implementing the wheels and body. The **Sedan** and **Truck** classes will each implement their own versions of the canopy. As you proceed through the following steps, you may wish to consult the **FacesApplication** project for examples of how to define an abstract class along with concrete classes that extend it.

1. Implement the **Vehicle** class:
   a. Define **Vehicle** to be a public abstract class. Provide the **Vehicle** class with protected instance variables to represent the body and wheels of a vehicle.
   b. Define a public **Vehicle** constructor that takes four integers (**left**, **top**, **width** and **height**) as inputs. These represent the location and dimensions of the body. The constructor should invoke the **makeObjects** method, with suitable parameters, to make the body, wheels and canopy and save them in instance variables. The constructor should also invoke the **addObjects** method to add these objects to the scene.
   c. Define a private **makeObjects** method that takes four integers (**left**, **top**, **width** and **height**) and calls **makeBody**, **makeWheel** (twice) and **makeCanopy** to make and store these component objects. Also define a private **addObjects** method (with no parameters) to add the vehicle components to the scene.
   d. Define two private methods to make vehicle components:
      i. **GObject makeBody(int left, int top, int width, int height)**
      ii. **GObject makeWheel(int left, int top, int diameter)**
e. Declare but do not implement a protected abstract helper method `makeCanopy` to take four integer parameters takes four integers (left, top, width and height) and return a `GObject` for the canopy of the car.

2. Implement the **Truck** class:
   a. Define **Truck** to be a public (non-abstract) class that extends **Vehicle**.
   b. Define a **Truck** constructor that takes four integers (left, top, width and height) as inputs. These represent the location and dimensions of the body. The constructor should simply invoke the **Vehicle** constructor with appropriate arguments.
   c. Define a protected method `makeCanopy` with the same signature as the `makeCanopy` method in the **Vehicle** class, except that the **Truck** class method is not abstract. Arrange for this method to return a rectangle representing the truck’s canopy.

3. Implement the **Sedan** class:
   a. Define **Sedan** to be a public (non-abstract) class that extends **Vehicle**.
   b. Define a **Sedan** constructor that takes four integers (left, top, width and height) as inputs. These represent the location and dimensions of the body. The constructor should simply invoke the **Vehicle** constructor with appropriate arguments.
   c. Define a protected method `makeCanopy` with the same signature as the `makeCanopy` method in the **Vehicle** class, except that the **Sedan** class method is not abstract. Arrange for this method to return a polygon representing the truck’s canopy.