Interface des Programmes d'Application, UE INF302

Corrected paper

Master Informatique 1
Université Bordeaux 1
Vendredi 18 mai 2007

No document, except a dictionary, length 3 hours

Exercise - Iterator

In a class `Iterators`, write a static method `public static <E> Iterator<E> append(Iterator<? extends E> it1, Iterator<? extends E> it2)` which returns an iterator which is the concatenation of it1 and it2. The method remove() will throw UnsupportedOperationException.

```
public static <E> Iterator<E> append(final Iterator<? extends E> it1,
     final Iterator<? extends E> it2) {
    return new Iterator<E>() {
        private boolean first = true;

        public boolean hasNext() {
            return it1.hasNext() || it2.hasNext();
        }

        public E next() {
            if (it1.hasNext()) {
                return it1.next();
            } else {
                first = false;
                return it2.next();
            }
        }

        public void remove() {
            throw new UnsupportedOperationException();
        }
    };
}
```

Problem - Vehicle

Question 1 Modify the class `DefaultVehicle` in such a way that the instruction

```
System.out.print(v);
```

where `v` is an instance of `DefaultVehicle` will print on the standard output the following result :
`Vehicle (`x`, `y`), speed = `s`, direction = `d

```
public String toString() {
    return "Vehicle (" + position.getX() + "," + position.getY()
        + "), speed = " + speed + ", direction = " + direction;
    }
```

Question 2 Give an implementation of `VehicleWithAccelerator` using inheritance.

```
public class VehicleWithAccelerator extends DefaultVehicle {

    public VehicleWithAccelerator(double maxSpeed) {
        super(maxSpeed);
    }

    public void accelerate(double acceleration) {
        setSpeed(Math.min(0, getSpeed() + acceleration));
    }
}
```

Question 3 Give another implementation of `VehicleWithAccelerator` without using inheritance from `DefaultVehicle`, but delegation.

```
public class VehicleWithAccelerator implements Vehicle {
    private DefaultVehicle delegate;

    public VehicleWithAccelerator(double maxSpeed) {
        delegate = new DefaultVehicle(maxSpeed);
    }

    public double getDirection() {
        return delegate.getDirection();
    }

    public Point2D getPosition() {
        return delegate.getPosition();
```

```
    }

    public double getSpeed() {
        return delegate.getSpeed();
    }

    public void run(double time) {
        delegate.run(time);
    }

    public void setDirection(double direction) {
        delegate.setDirection(direction);
    }

    public void accelerate(double acceleration) {
        double oldSpeed = delegate.getSpeed();
        delegate.setSpeed(Math.min(0, oldSpeed + acceleration));
    }
}
```

Question 4 Implement a class `RectangleArea` which implements `Area`.

```
public class RectangleArea implements Area {

    private double x, y, width, height;

    public RectangleArea(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    private void testPoint(Point2D p) {
        if (! contains(p))
            throw new IllegalArgumentException(p +
                " does not belong to the rectangle");
    }

    public boolean areConnected(Point2D p1, Point2D p2) {
        testPoint(p1);
        testPoint(p2);
        return true;
    }
```

```
    public boolean contains(Point2D p) {
        return p.getX() >= x && p.getX() <= x + width &&
            p.getY() >= y && p.getY() <= y + height;
    }
}
```

Question 5 Create a class `AreaWithVehicle` which is composed of an area and a vehicle, given at the construction.

```
public class AreaWithVehicle {

    private final Area area;
    private final Vehicle vehicle;

    public AreaWithVehicle(Area area, Vehicle vehicle) {
        this.area = area;
        this.vehicle = vehicle;
    }

    public void runVehicle(double time) throws AccidentException {
        Point2D position;

        vehicle.run(time);
        position = vehicle.getPosition();
        if (!area.contains(position))
            throw new AccidentException(position);
    }
}
```

Question 6 Propose an implementation for the exception `AccidentException`.

```
public class AccidentException extends Exception {
    Point2D position;

    public AccidentException(Point2D position) {
      this.position = position;
    }

    public Point2D position() {
      return position;
    }
}
```

Question 7 Implement a class `ObservableVehicle` which extends `Observable` and implements `Vehicle`. Use the class `VehicleWithAccelerator` by delegation to implement the methods of `Vehicle`.

Each time an observable vehicle moves, its observers must be notified. Take care that the method `setChanged()` must always be called before using `notifyObservers`.

```
public class ObservableVehicle extends Observable implements Vehicle {
    private Vehicle vehicle;

    public ObservableVehicle(double maxSpeed) {
      vehicle = new VehicleWithAccelerator(maxSpeed);
    }

    public double getDirection() {
      return vehicle.getDirection();
    }

    public Point2D getPosition() {
      return vehicle.getPosition();
    }

    public double getSpeed() {
      return vehicle.getSpeed();
    }

    public void run(double time) {
      vehicle.run(time);
      setChanged();
      notifyObservers();
    }
}
```

Question 8 Propose a new version of the class `AreaWithVehicle` which will be constructed using an instance of `ObservableVehicle`.

```
public class AreaWithVehicle implements Observer {

    private final Area area;
    private final Vehicle vehicle;

    public AreaWithVehicle(Area area, ObservableVehicle vehicle) {
        this.area = area;
```

```
        this.vehicle = vehicle;
        vehicle.addObserver(this);
    }

    public void update(Observable o, Object arg) {
        Point2D position = vehicle.getPosition();
        if (!area.contains(position))
            throw new AccidentException(position);
    }
}
```

Moreover, `AccidentException` must inherit from `RuntimeException` because it cannot be declared in the method `update`.

A better solution would be to throw a `RuntimeException` in the `update` method, to catch it in the method `run` in `ObservableVehicle` and transform it into an `AccidentException`.

```
public class AreaWithVehicle implements Observer {
    ...
    public void update(Observable o, Object arg) {
        Point2D position = vehicle.getPosition();
        if (!area.contains(position))
            throw new RuntimeException();
    }
}

public class ObservableVehicle extends Observable implements Vehicle {
    ...
    public void run(double time) throws AccidentException {
        vehicle.run(time);
        setChanged();
        try {
            notifyObservers();
        } catch (Exception e) {
            throw new AccidentException(getPosition());
        }
    }
}
```