```java
//********************************************************************************
//
// File:    Link.java
// Package: ---
// Unit:    Class Link
//
//********************************************************************************

import edu.rit.sim.Simulation;

/**
 * Class Link represents a connection between two routable objects. Links are
 * <I>closed</I> if a packet is currently transmitting on them and <I>open</I>
 * if they are ready to be transmitted on.
 *
 * @author Jimi Ford (jhf3617)
 * @version 5-6-2015
 */
public class Link {

    /**
     * default bit rate used in this project
     * <P>9600 bits/sec</P>
     */
    public static final int DEFAULT_BIT_RATE = 9600;

    /**
     * true if this link has an infinite bit rate
     */
    public final boolean infiniteBitRate;

    /**
     * the bit rate of this link
     */
    public final double bitRate;

    // private data members

    private final Routable r1;
    private final Routable r2;
    private final Simulation sim;
    private double closeStarted;
    private double closeFinished;
    private double totalTimeSpentClosed;
    private boolean ready;


    /**
     * construct a link with the default finit bit rate between two routables
     *
     * @param sim the simulation reference
     * @param r1 one of the routable objects
     * @param r2 the other routable object
     */
    public Link(Simulation sim, Routable r1, Routable r2) {
        this(sim, false, r1, r2);
    }

```

```java
    /**
     * construct a link with specified finite or infinite bit rate
     *
     * @param sim the simulation reference
     * @param infiniteBitRate set to true for infinite bit rate, false for
     * default finite bit rate
     * @param r1 one of the routable objects
     * @param r2 the other routable object
     */
    public Link(Simulation sim, boolean infiniteBitRate, Routable r1,
            Routable r2) {
        this.sim = sim;
        this.r1 = r1;
        this.r2 = r2;
        this.ready = true;
        this.infiniteBitRate = infiniteBitRate;
        this.bitRate = infiniteBitRate ? Double.POSITIVE_INFINITY :
            DEFAULT_BIT_RATE;
        this.totalTimeSpentClosed = 0;
    }

    /**
     * get the other routable object attached to this link compared to the
     * current one
     *
     * @param current the current routable object querying for the other
     * attached routable object
     * @return the routable object that is not equal to the current one
     */
    public Routable other(Routable current) {
        return this.r1.equals(current) ? r2 : r1;
    }

    /**
     * get the current state of the link
     *
     * @return true if the link is ready to pass another packet along it; false
     * otherwise
     */
    public boolean ready() {
        return this.ready;
    }

    /**
     * close this link off so that other packets may not be transmitted on it
     * until open() is called
     *
     * @throws IllegalStateException if the link is not ready to be closed and
     * this link has a finite bit-rate
     */
    public void close() throws IllegalStateException {
        if(!this.infiniteBitRate) {
            if(!this.ready) {
                throw new IllegalStateException();
            }
            this.ready = false;
            this.closeStarted = sim.time();
        }
```

```
117     }
118
119     /**
120      * open this link so that other packets may be transmitted on it
121      */
122     public void open() {
123         this.ready = true;
124         this.closeFinished = sim.time();
125         this.totalTimeSpentClosed += (this.closeFinished - this.closeStarted);
126     }
127
128     /**
129      * Return the amount of time this link was closed as a fraction of the
130      * total amount of time in the simulation.
131      */
132     public double fractionClosed() {
133         return this.totalTimeSpentClosed / sim.time();
134     }
135 }
136
```