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

import edu.rit.sim.Event;
import edu.rit.sim.Simulation;

/**
 * Class Routable is the abstract base class that defines objects that contain
 * routing logic with the ability to be linked together. Known implementations
 * include Router and Host.
 *
 * @author Jimi Ford (jhf3617)
 * @version 5-6-2015
 */
public abstract class Routable {

    private static int count = 0;

    /**
     * the simulation reference
     */
    protected final Simulation sim;
    private final int id;


    /**
     * Construct a routable object
     * @param sim the simulation this object should belong to
     */
    public Routable(Simulation sim) {
        this.sim = sim;
        id = ++ count;
    }

    /**
     * compare this instance with another object and determine whether this
     * instance is equal to the other object.
     *
     * @param o the other object to compare to
     * @return true if this object is equal to the other object
     */
    public boolean equals(Object o) {
        if(o == this) {
            return true;
        }
        if(o instanceof Routable) {
            return this.id == ((Routable)o).id;
        }
        return false;
    }

    /**
     * Called when this routable object finished receiving a packet on a certain
     * link
     * @param packet the packet this object received
     * @param link the link that the packet was received on
     */
    public abstract void receivePacket(final Packet packet, final Link link);

    /**
     * Send a given packet along a given link to another routable
     *
     * @param packet the packet to send
     * @param link the link to send the packet along
     */
    public void startSending(final Packet packet, final Link link) {
        final Routable other = link.other(this);
        final double transmitTime = packet.transmitTime(link);
        link.close();
        sim.doAfter(transmitTime, new Event() {
            public void perform() {
                other.receivePacket(packet, link);
            }
        });
    }
}

```