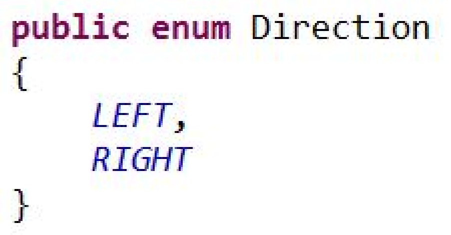
Interfaces and Inheritance

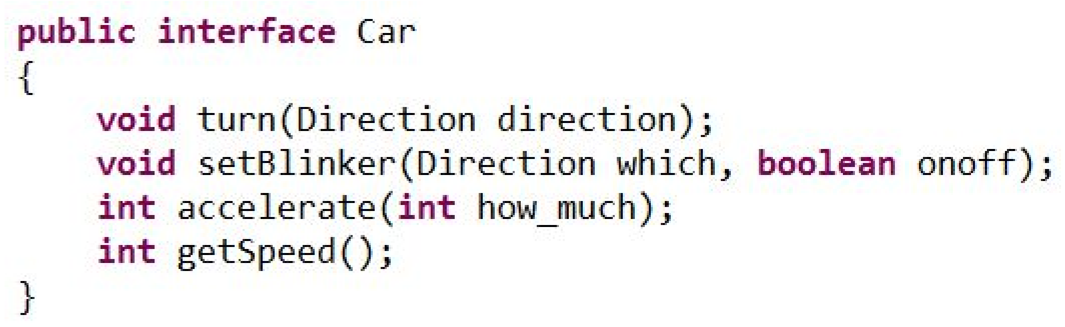
Michael Wagner

Recall: Enumeration



```
public enum Direction
{
    LEFT,
    RIGHT
}
```

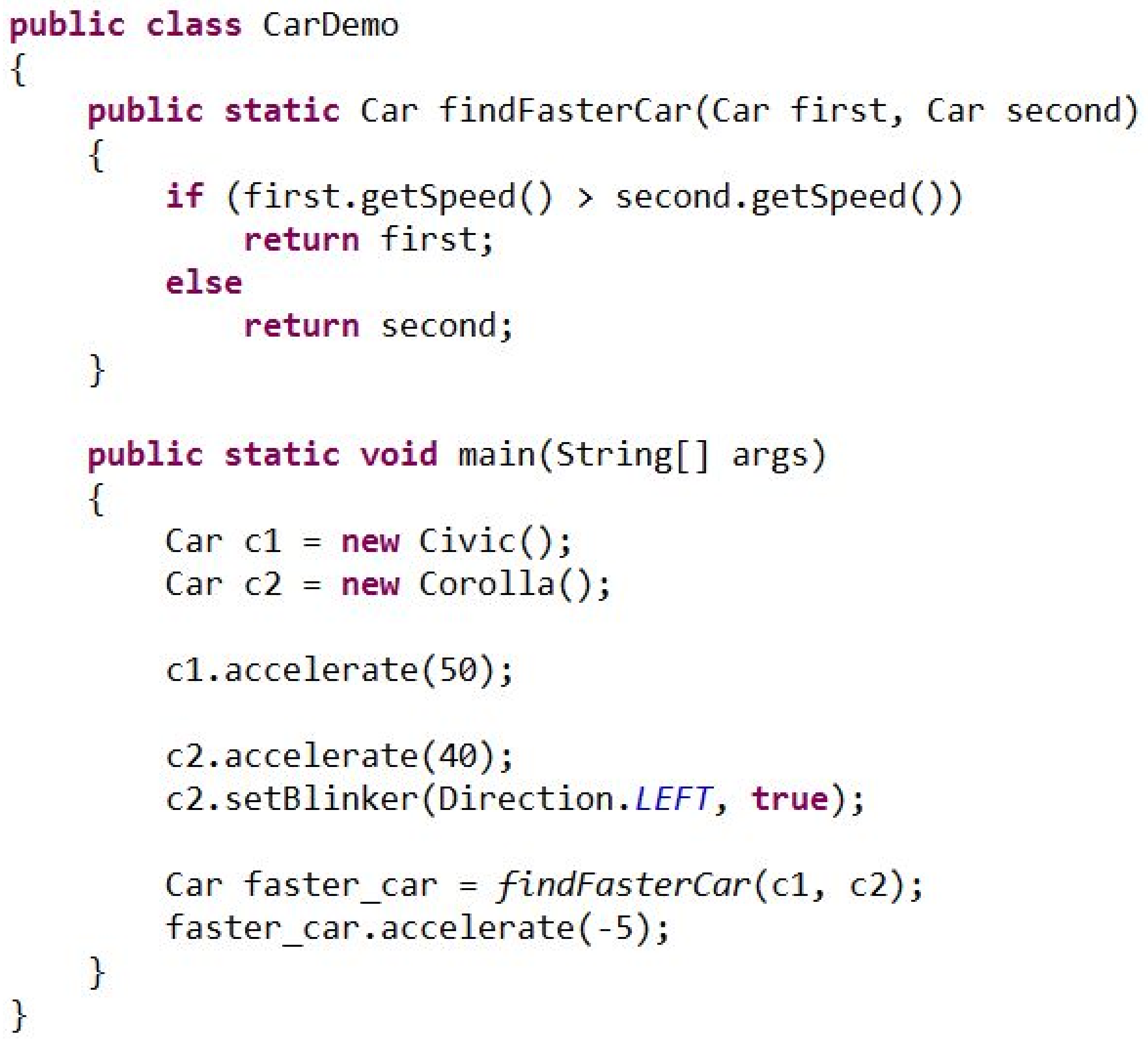
Interface



```java
public interface Car
{
    void turn(Direction direction);
    void setBlinker(Direction which, boolean onoff);
    int accelerate(int how_much);
    int getSpeed();
}
```

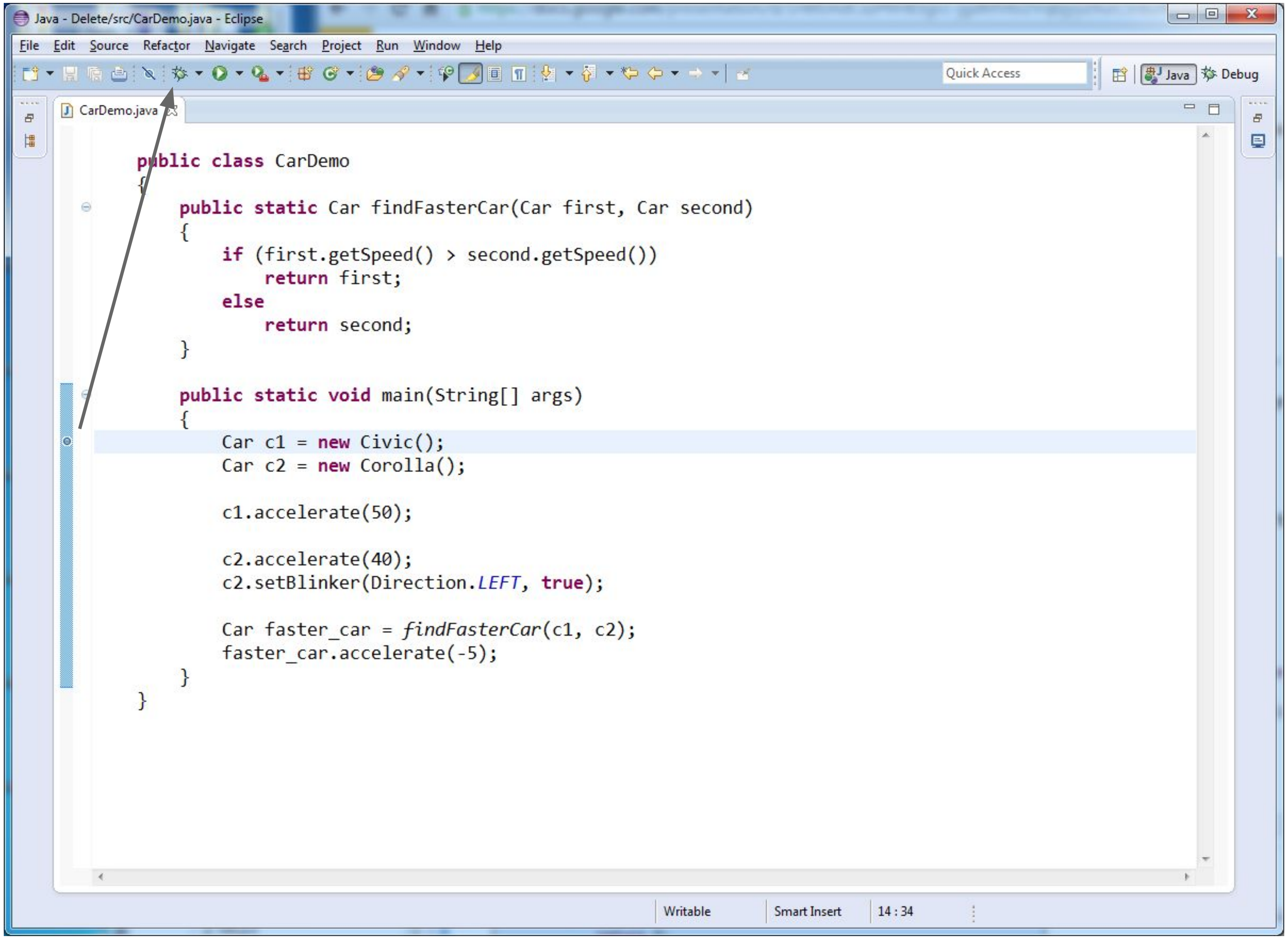
```java
public class Civic implements Car
{
    private int speed = 0;

    private boolean left_blinker_on = false;
    private boolean right_blinker_on = false;

    public void turn(Direction direction)
    {
        setBlinker(direction, true);
        accelerate(-15);
    }

    public void setBlinker(Direction which, boolean onoff)
    {
        if (which == Direction.LEFT)
        {
            left_blinker_on = onoff;
        }
        else if (which == Direction.RIGHT)
        {
            right_blinker_on = onoff;
        }
    }

    public int accelerate(int how_much)
    {
        speed += how_much;
        if (speed > 100)
            speed = 100;
        return getSpeed();
    }

    public int getSpeed()
    {
        return speed;
    }
}
```

Interface is a Type



```java
public class CarDemo
{
    public static Car findFasterCar(Car first, Car second)
    {
        if (first.getSpeed() > second.getSpeed())
            return first;
        else
            return second;
    }

    public static void main(String[] args)
    {
        Car c1 = new Civic();
        Car c2 = new Corolla();

        c1.accelerate(50);

        c2.accelerate(40);
        c2.setBlinker(Direction.LEFT, true);

        Car faster_car = findFasterCar(c1, c2);
        faster_car.accelerate(-5);
    }
}
```

Debugger


Java - Delete/src/CarDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
CarDemo.java
Writable
Smart Insert
14 : 34

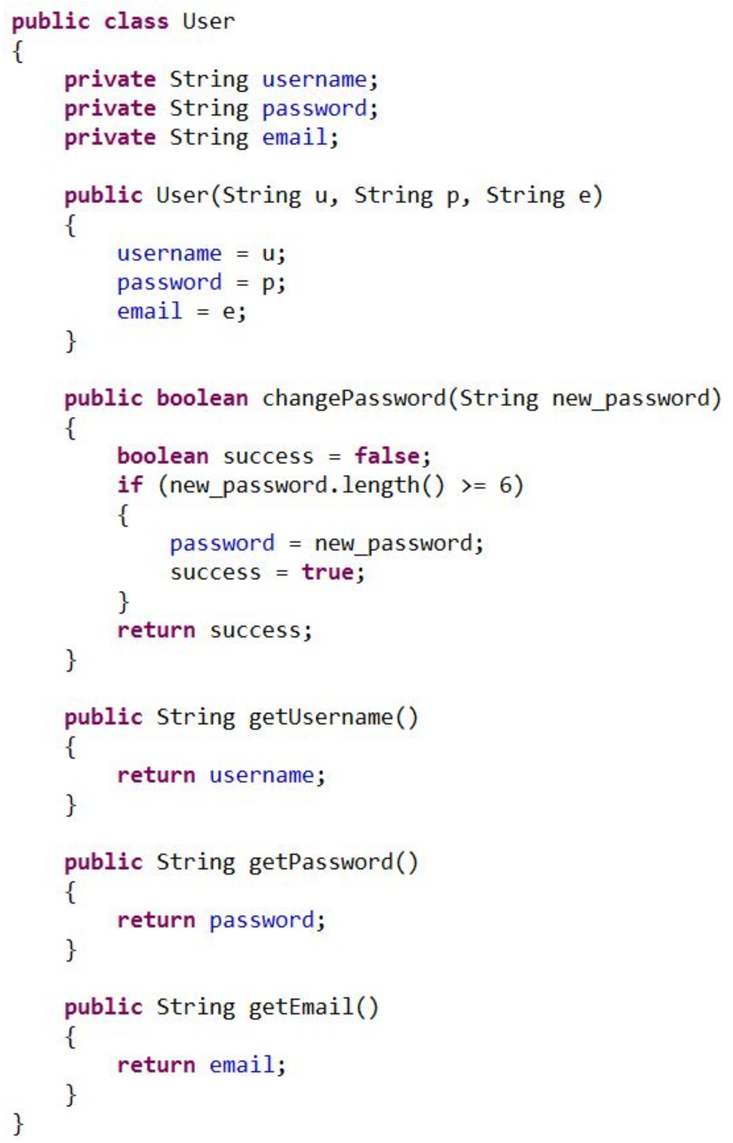
```
public class CarDemo
{
    public static Car findFasterCar(Car first, Car second)
    {
        if (first.getSpeed() > second.getSpeed())
            return first;
        else
            return second;
    }

    public static void main(String[] args)
    {
        Car c1 = new Civic();
        Car c2 = new Corolla();

        c1.accelerate(50);

        c2.accelerate(40);
        c2.setBlinker(Direction.LEFT, true);

        Car faster_car = findFasterCar(c1, c2);
        faster_car.accelerate(-5);
    }
}
```

Example

Add the **Product** class and **Relatable** interface to a project. Make **Product** implement **Relatable**. Create a **ProductDemo** that has a method with signature:

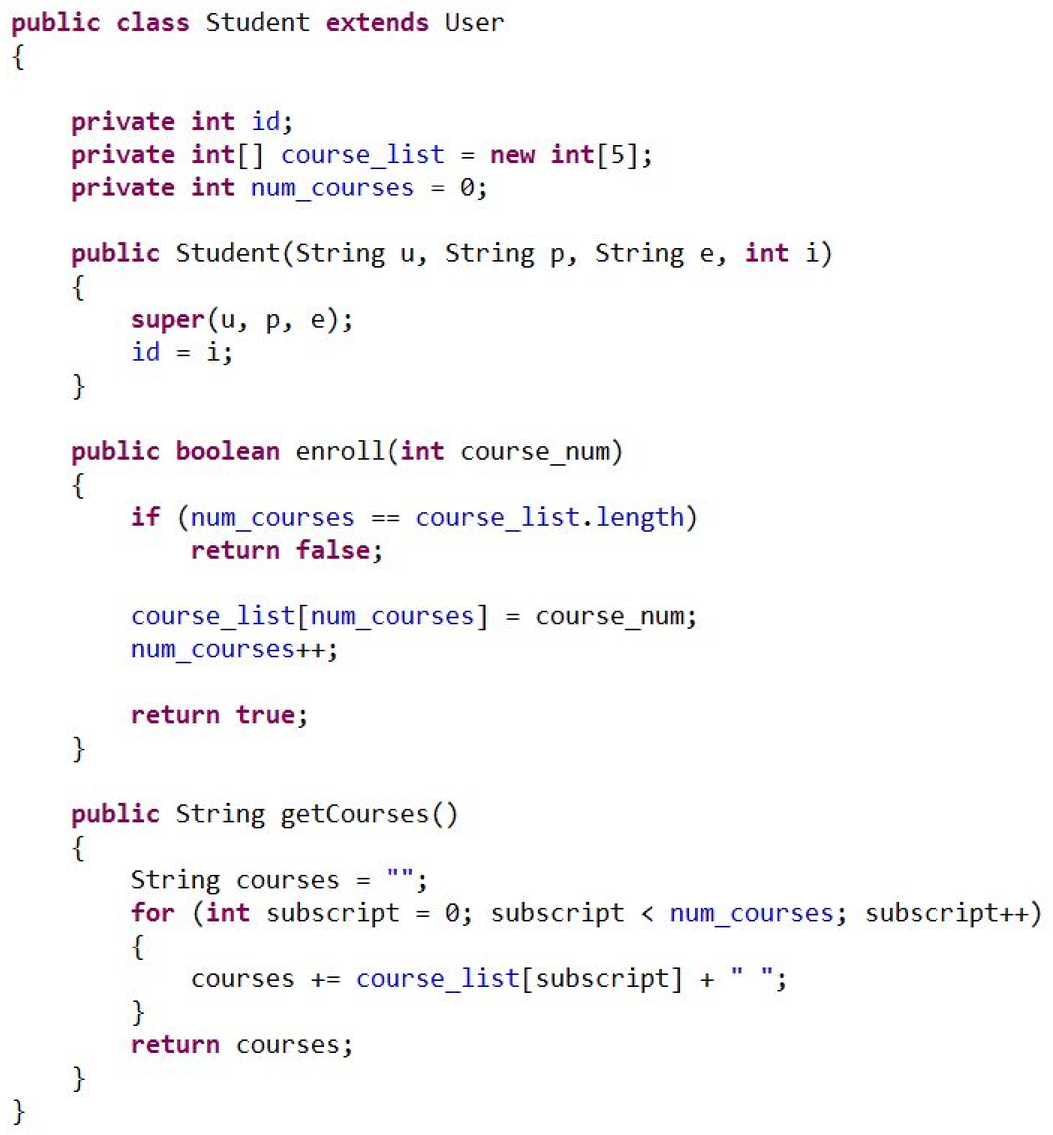
- **public static Object findCheapest(Relatable[] arr)**

Inheritance



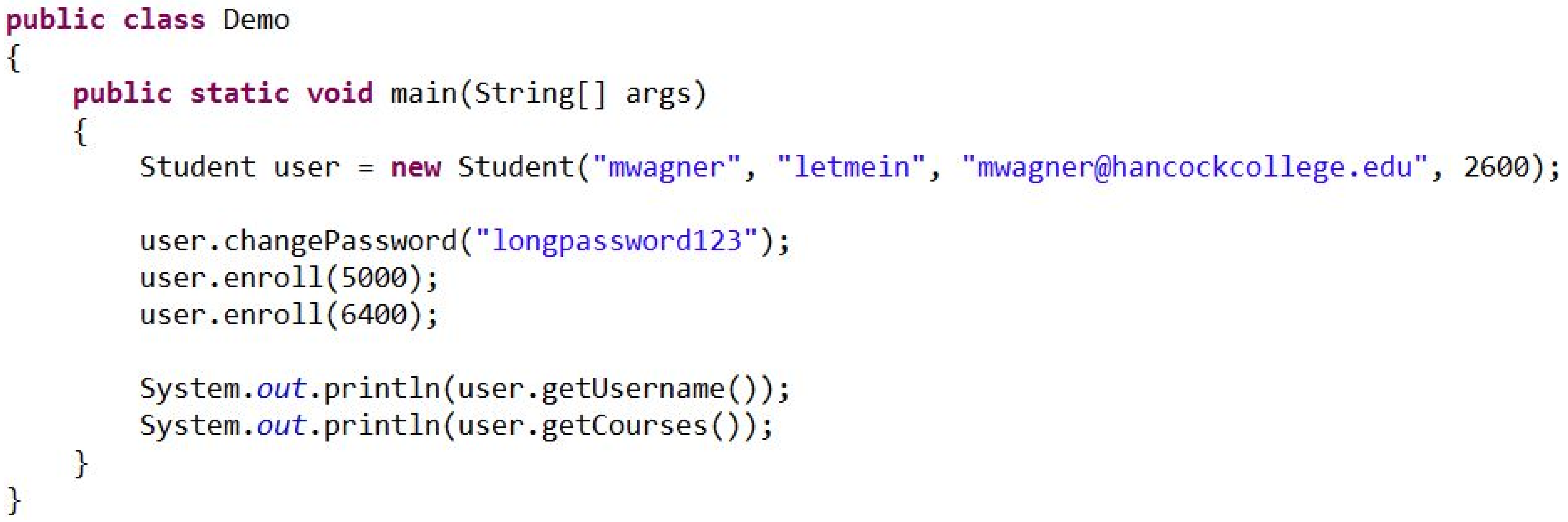
```
public class User
{
    private String username;
    private String password;
    private String email;

    public User(String u, String p, String e)
    {
        username = u;
        password = p;
        email = e;
    }

    public boolean changePassword(String new_password)
    {
        boolean success = false;
        if (new_password.length() >= 6)
        {
            password = new_password;
            success = true;
        }
        return success;
    }

    public String getUsername()
    {
        return username;
    }

    public String getPassword()
    {
        return password;
    }

    public String getEmail()
    {
        return email;
    }
}
```

Inheritance - Subclass



```java
public class Student extends User
{

    private int id;
    private int[] course_list = new int[5];
    private int num_courses = 0;

    public Student(String u, String p, String e, int i)
    {
        super(u, p, e);
        id = i;
    }

    public boolean enroll(int course_num)
    {
        if (num_courses == course_list.length)
            return false;

        course_list[num_courses] = course_num;
        num_courses++;

        return true;
    }

    public String getCourses()
    {
        String courses = "";
        for (int subscript = 0; subscript < num_courses; subscript++)
        {
            courses += course_list[subscript] + " ";
        }
        return courses;
    }
}
```

Inheritance - Usage



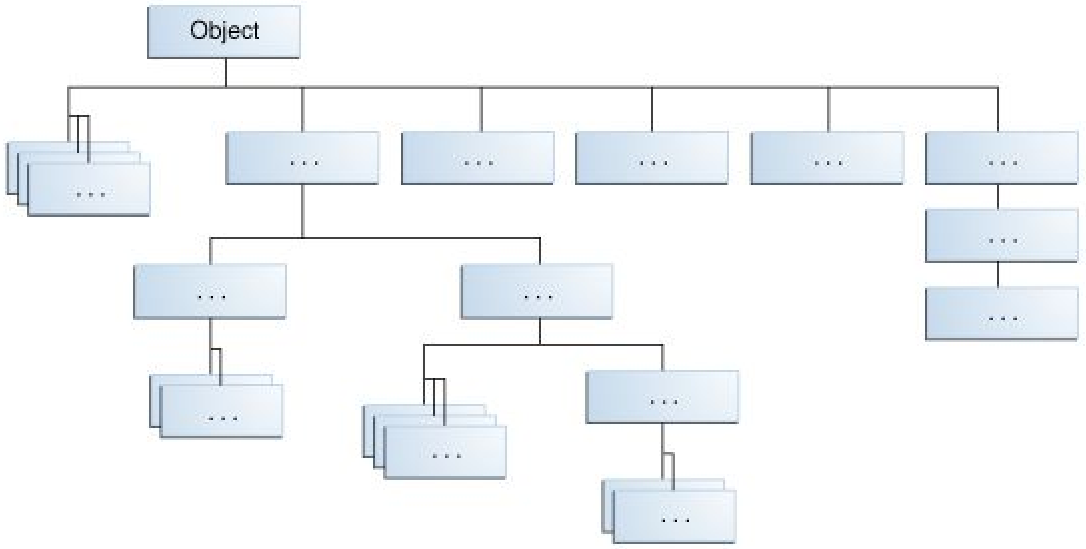
```java
public class Demo
{
    public static void main(String[] args)
    {
        Student user = new Student("mwagner", "letmein", "mwagner@hancockcollege.edu", 2600);

        user.changePassword("longpassword123");
        user.enroll(5000);
        user.enroll(6400);

        System.out.println(user.getUsername());
        System.out.println(user.getCourses());
    }
}
```

Example

Create a class called **Redditor** that is a subclass of **User**. Make the following methods:

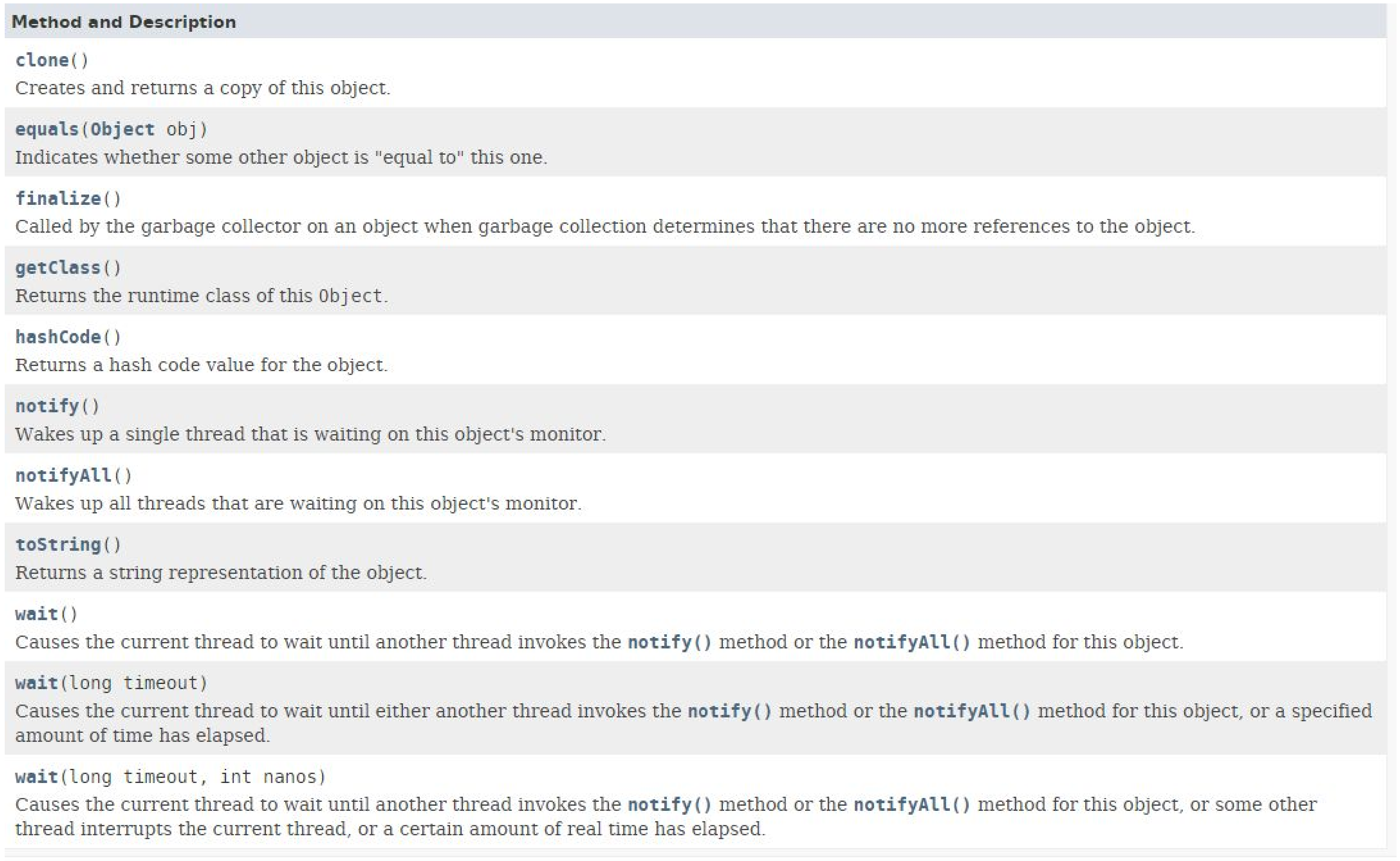
- boolean postLink(String title, String url)
    - Make an inner class that encapsulates title-url
    - Use an ArrayList to store posts
- Use a static member to prevent reposts
- Create a static method to get all unique posts

Object



https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

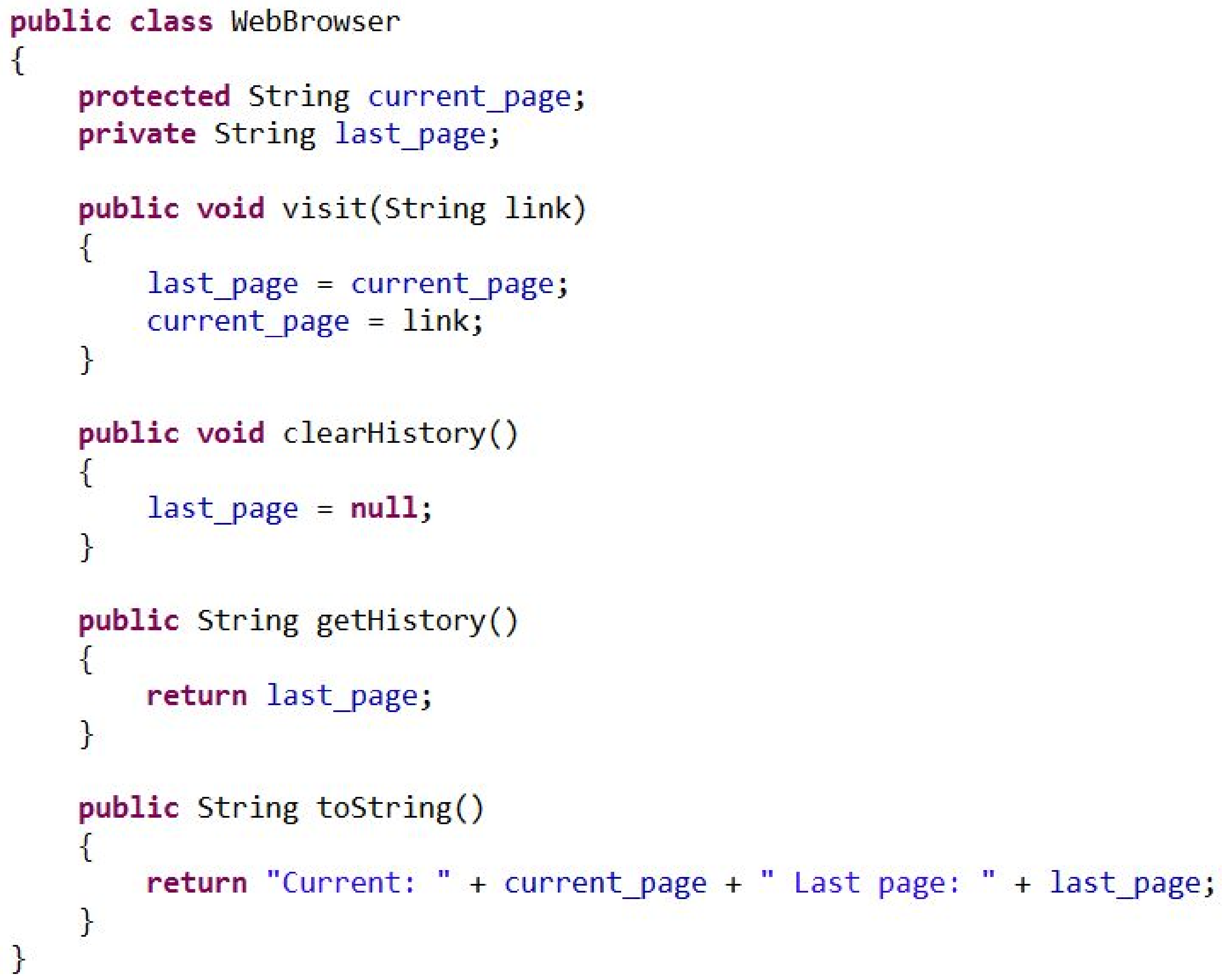
Object



Method and Description
clone() Creates and returns a copy of this object.
equals(**Object** obj) Indicates whether some other object is "equal to" this one.
finalize() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
getClass() Returns the runtime class of this `Object`.
hashCode() Returns a hash code value for the object.
notify() Wakes up a single thread that is waiting on this object's monitor.
notifyAll() Wakes up all threads that are waiting on this object's monitor.
toString() Returns a string representation of the object.
wait() Causes the current thread to wait until another thread invokes the **notify()** method or the **notifyAll()** method for this object.
wait(long timeout) Causes the current thread to wait until either another thread invokes the **notify()** method or the **notifyAll()** method for this object, or a specified amount of time has elapsed.
wait(long timeout, int nanos) Causes the current thread to wait until another thread invokes the **notify()** method or the **notifyAll()** method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

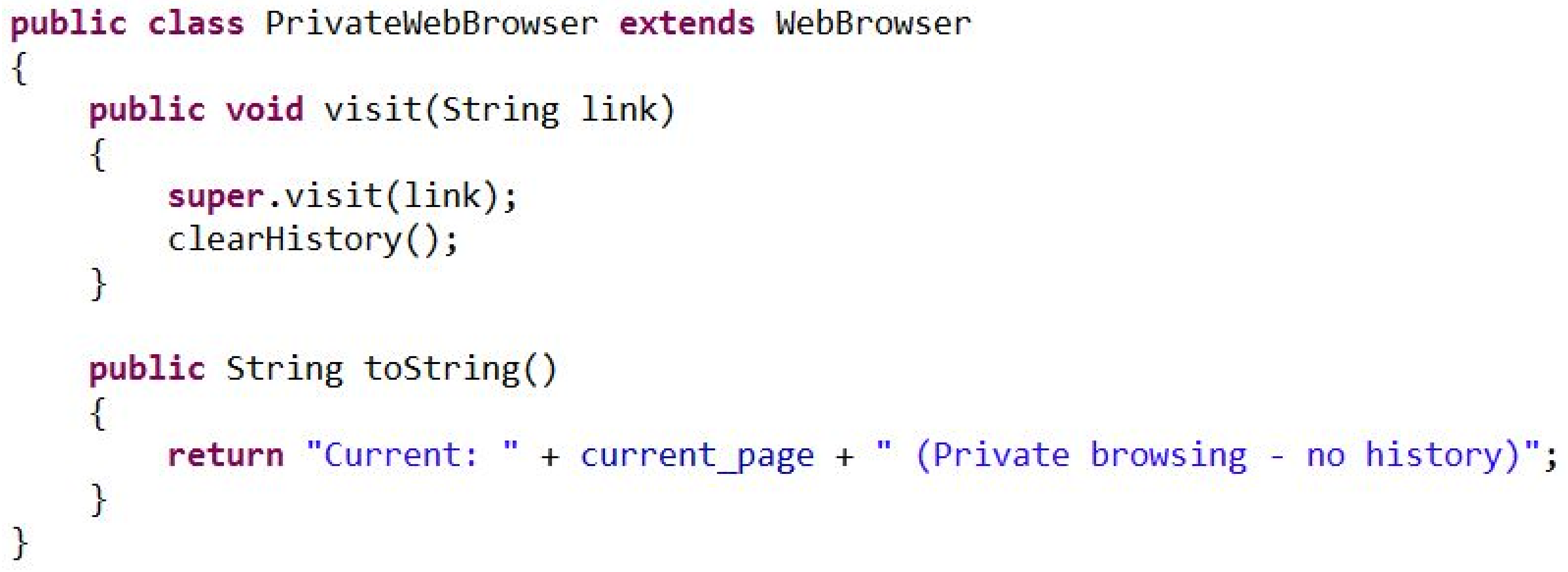
https://docs.oracle.com/javase/8/docs/api/

Overriding - Superclass



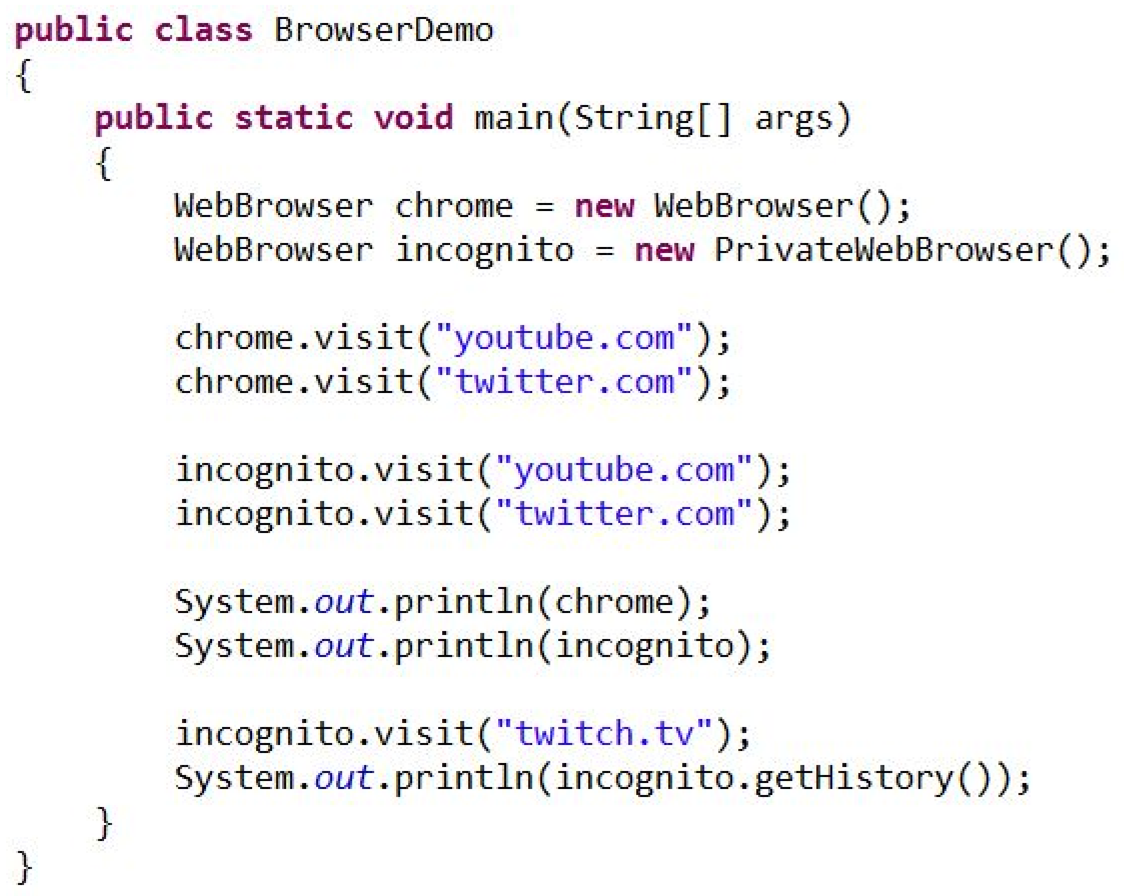
```
public class WebBrowser
{
    protected String current_page;
    private String last_page;

    public void visit(String link)
    {
        last_page = current_page;
        current_page = link;
    }

    public void clearHistory()
    {
        last_page = null;
    }

    public String getHistory()
    {
        return last_page;
    }

    public String toString()
    {
        return "Current: " + current_page + " Last page: " + last_page;
    }
}
```

Overriding - Subclass



```java
public class PrivateWebBrowser extends WebBrowser
{
    public void visit(String link)
    {
        super.visit(link);
        clearHistory();
    }

    public String toString()
    {
        return "Current: " + current_page + " (Private browsing - no history)";
    }
}
```

Overriding Demo



```java
public class BrowserDemo
{
    public static void main(String[] args)
    {
        WebBrowser chrome = new WebBrowser();
        WebBrowser incognito = new PrivateWebBrowser();

        chrome.visit("youtube.com");
        chrome.visit("twitter.com");

        incognito.visit("youtube.com");
        incognito.visit("twitter.com");

        System.out.println(chrome);
        System.out.println(incognito);

        incognito.visit("twitch.tv");
        System.out.println(incognito.getHistory());
    }
}
```

Example

Make a class called **Message**.

- String member and getter

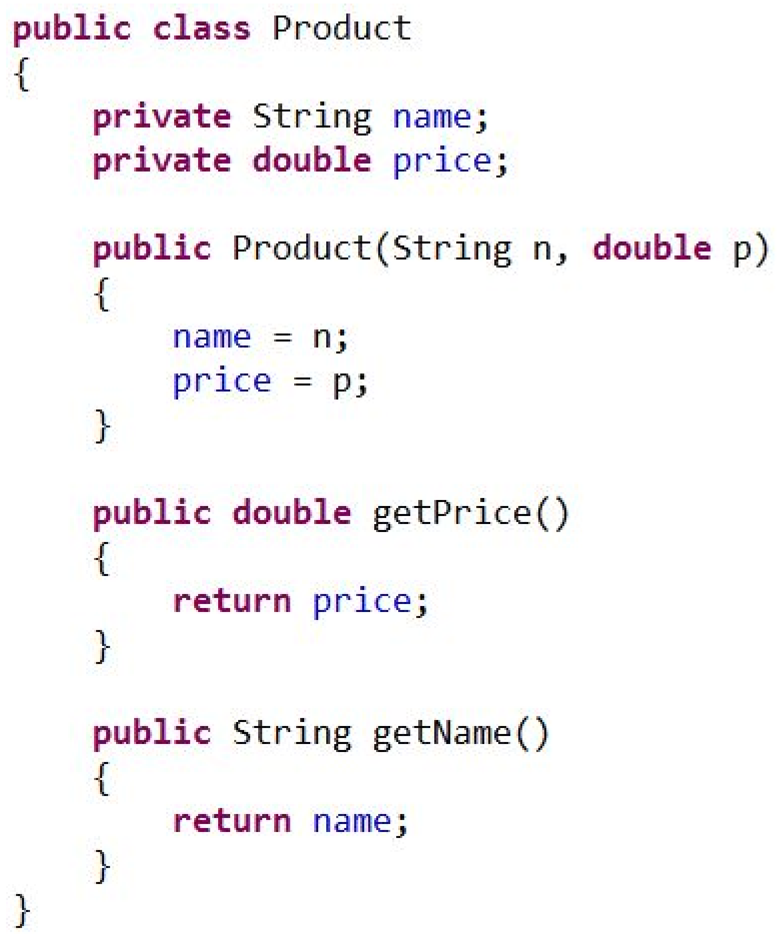
Make a subclass called **TextMessage**

- Override message getter to return only 160 characters

Make a subclass called **StatusUpdate**

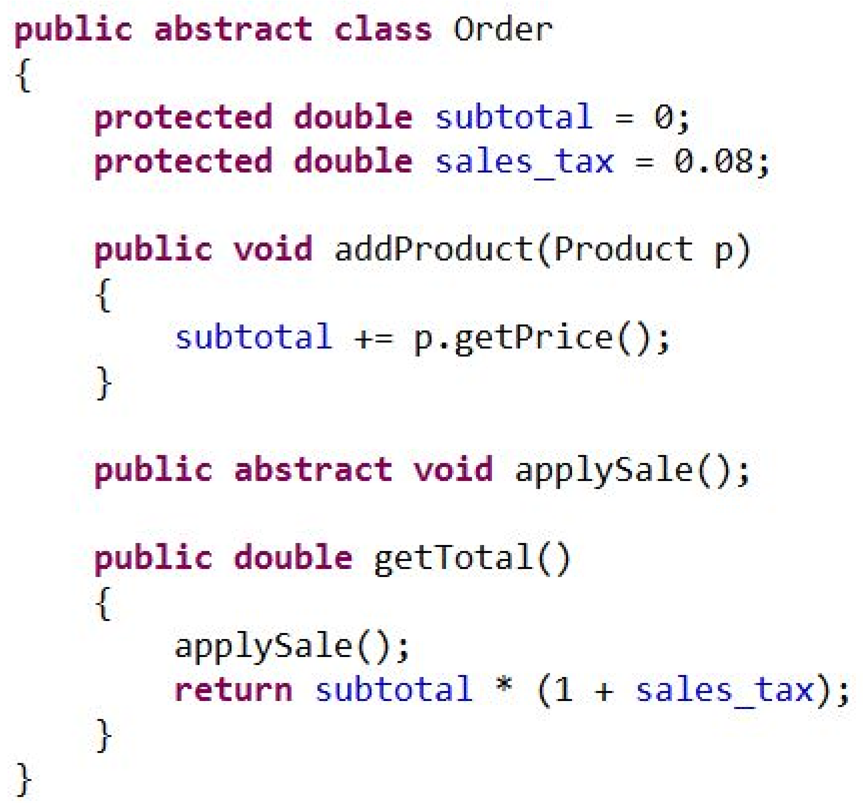
- Add a **like** method
- Override message to get message and like count

For next slide



```java
public class Product
{
    private String name;
    private double price;

    public Product(String n, double p)
    {
        name = n;
        price = p;
    }

    public double getPrice()
    {
        return price;
    }

    public String getName()
    {
        return name;
    }
}
```

Abstract Class



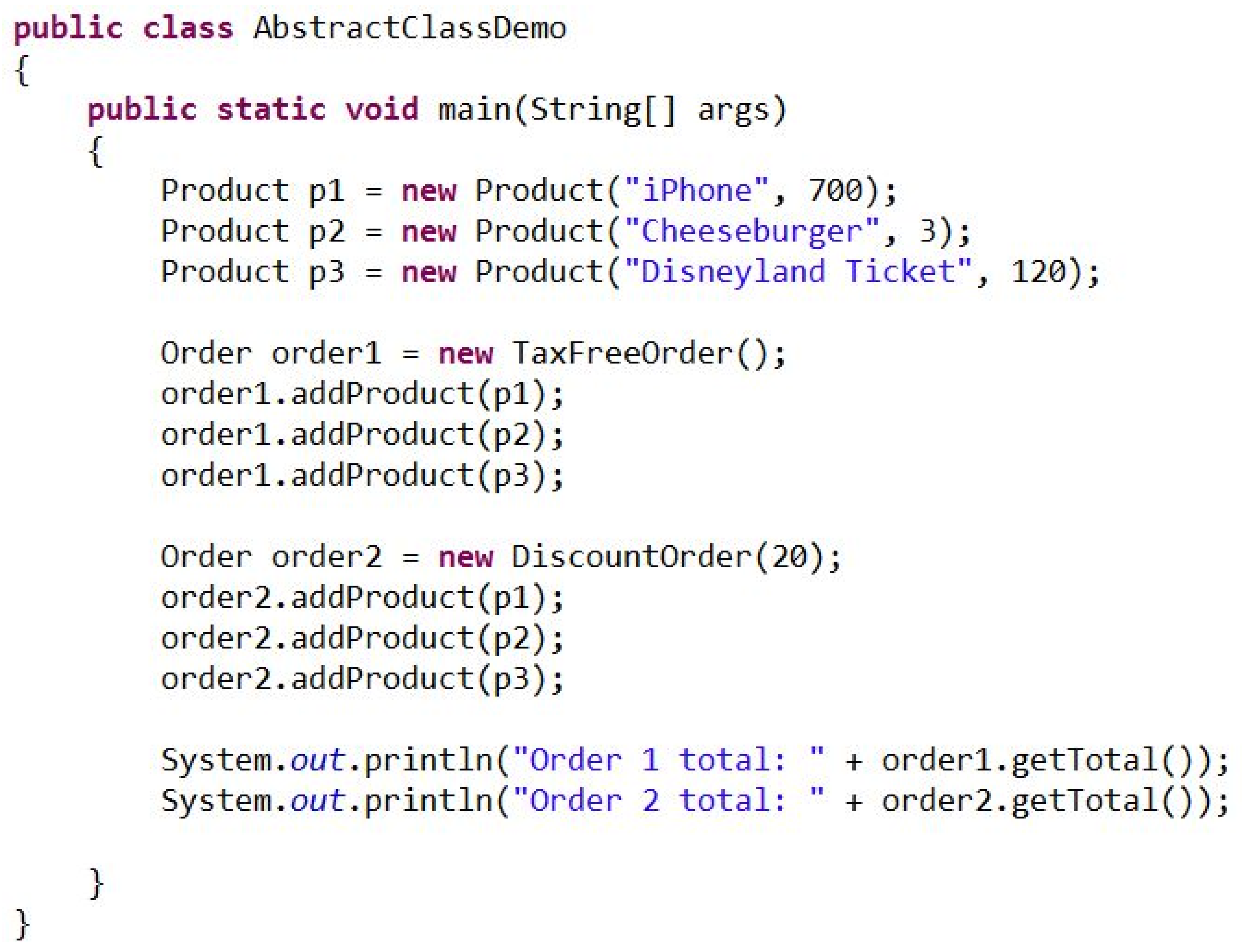
```
public abstract class Order
{
    protected double subtotal = 0;
    protected double sales_tax = 0.08;

    public void addProduct(Product p)
    {
        subtotal += p.getPrice();
    }

    public abstract void applySale();

    public double getTotal()
    {
        applySale();
        return subtotal * (1 + sales_tax);
    }
}
```

Abstract Class - Subclasses

```
public class TaxFreeOrder extends Order
{
    public void applySale()
    {
        sales_tax = 0;
    }
}
```

```
public class DiscountOrder extends Order
{
    private double off;

    public DiscountOrder(double off)
    {
        this.off = off;
    }

    public void applySale()
    {
        subtotal -= off;
    }
}
```

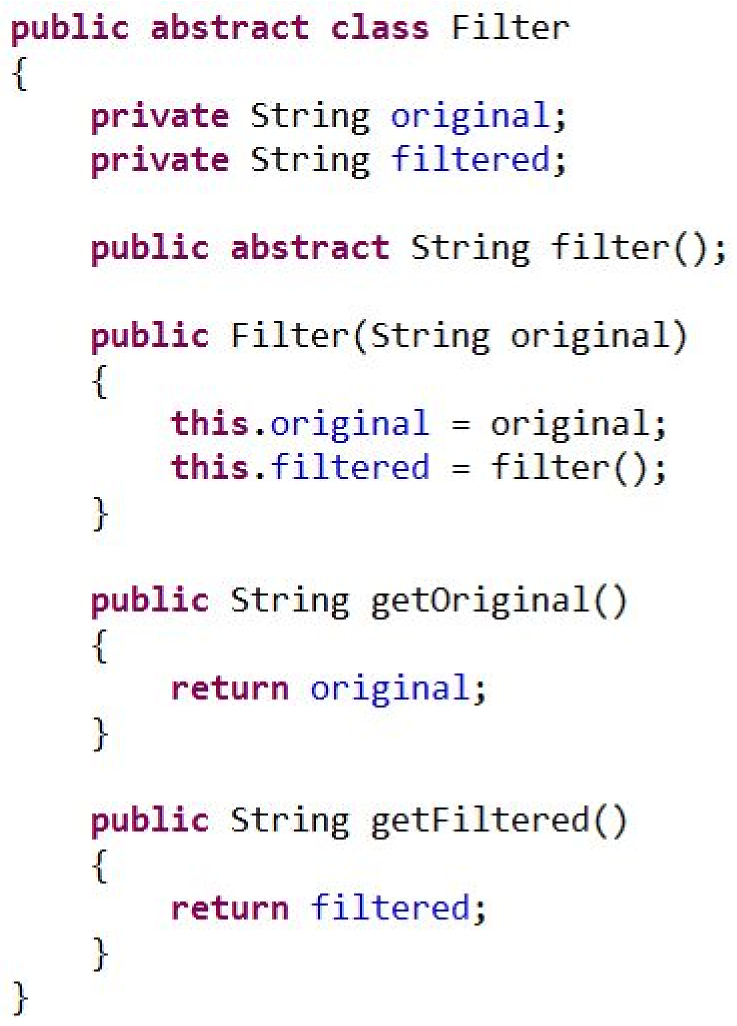
Abstract Class Demo



```java
public class AbstractClassDemo
{
    public static void main(String[] args)
    {
        Product p1 = new Product("iPhone", 700);
        Product p2 = new Product("Cheeseburger", 3);
        Product p3 = new Product("Disneyland Ticket", 120);

        Order order1 = new TaxFreeOrder();
        order1.addProduct(p1);
        order1.addProduct(p2);
        order1.addProduct(p3);

        Order order2 = new DiscountOrder(20);
        order2.addProduct(p1);
        order2.addProduct(p2);
        order2.addProduct(p3);

        System.out.println("Order 1 total: " + order1.getTotal());
        System.out.println("Order 2 total: " + order2.getTotal());

    }
}
```

Example

Create an abstract class called **Filter**. Create a subclass called **BadWordFilter**.



```java
public abstract class Filter
{
    private String original;
    private String filtered;

    public abstract String filter();

    public Filter(String original)
    {
        this.original = original;
        this.filtered = filter();
    }

    public String getOriginal()
    {
        return original;
    }

    public String getFiltered()
    {
        return filtered;
    }
}
```