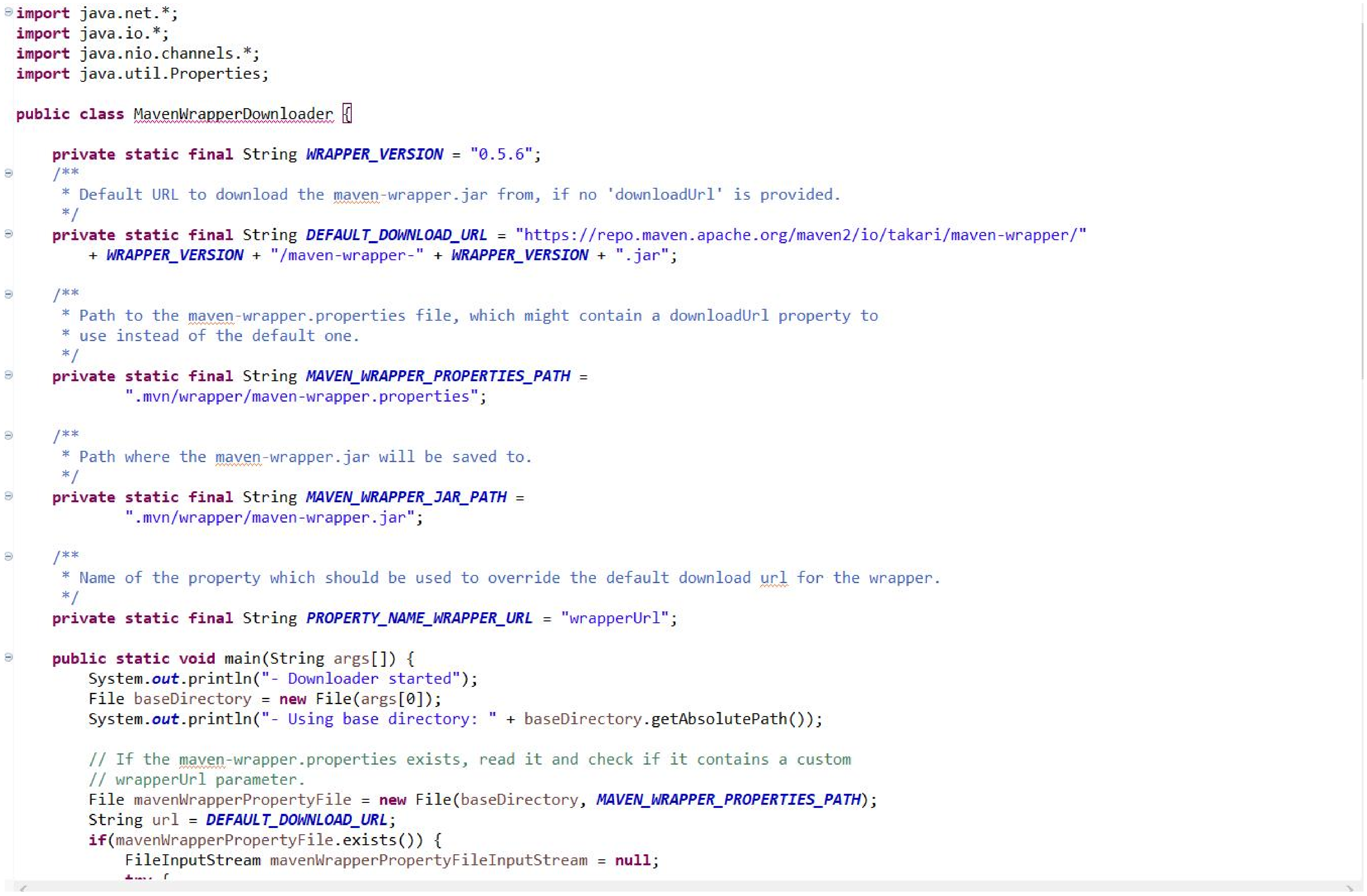
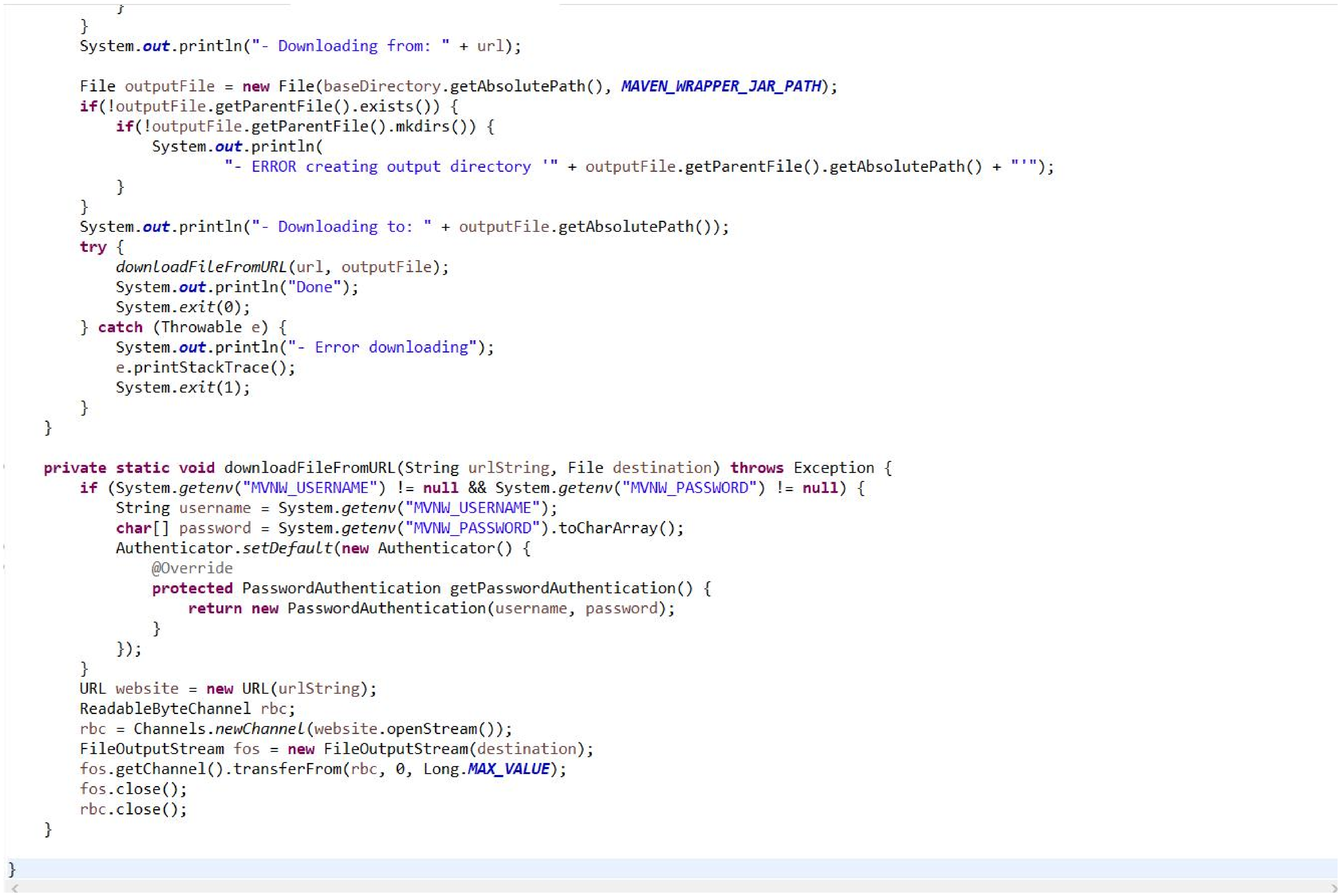
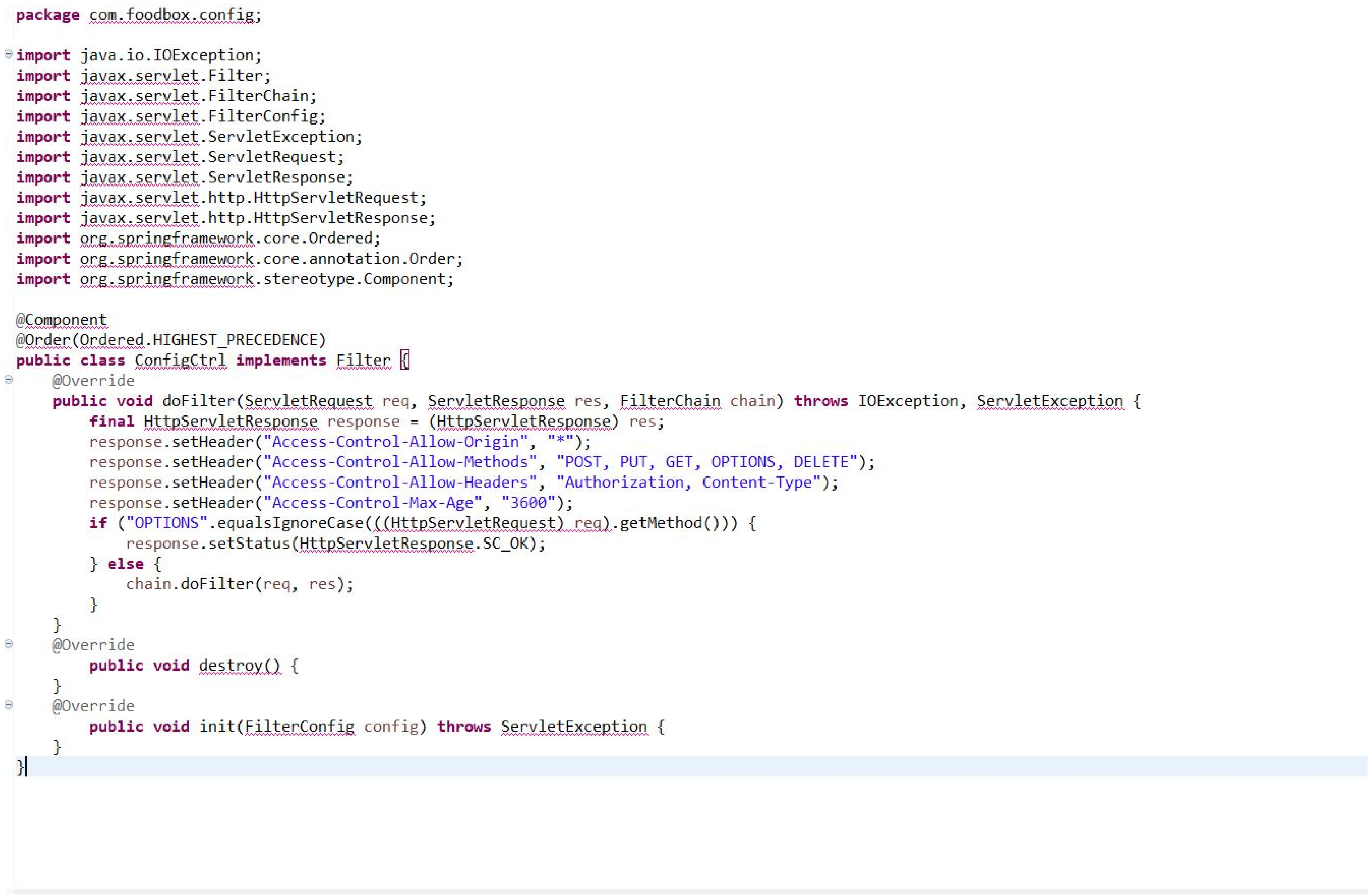


```java
import java.net.*;
import java.io.*;
import java.nio.channels.*;
import java.util.Properties;

public class MavenWrapperDownloader {

    private static final String WRAPPER_VERSION = "0.5.6";
    /**
     * Default URL to download the maven-wrapper.jar from, if no 'downloadUrl' is provided.
     */
    private static final String DEFAULT_DOWNLOAD_URL = "https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/"
        + WRAPPER_VERSION + "/maven-wrapper-" + WRAPPER_VERSION + ".jar";

    /**
     * Path to the maven-wrapper.properties file, which might contain a downloadUrl property to
     * use instead of the default one.
     */
    private static final String MAVEN_WRAPPER_PROPERTIES_PATH =
            ".mvn/wrapper/maven-wrapper.properties";

    /**
     * Path where the maven-wrapper.jar will be saved to.
     */
    private static final String MAVEN_WRAPPER_JAR_PATH =
            ".mvn/wrapper/maven-wrapper.jar";

    /**
     * Name of the property which should be used to override the default download url for the wrapper.
     */
    private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";

    public static void main(String args[]) {
        System.out.println("- Downloader started");
        File baseDirectory = new File(args[0]);
        System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());

        // If the maven-wrapper.properties exists, read it and check if it contains a custom
        // wrapperUrl parameter.
        File mavenWrapperPropertyFile = new File(baseDirectory, MAVEN_WRAPPER_PROPERTIES_PATH);
        String url = DEFAULT_DOWNLOAD_URL;
        if(mavenWrapperPropertyFile.exists()) {
            FileInputStream mavenWrapperPropertyFileInputStream = null;
```

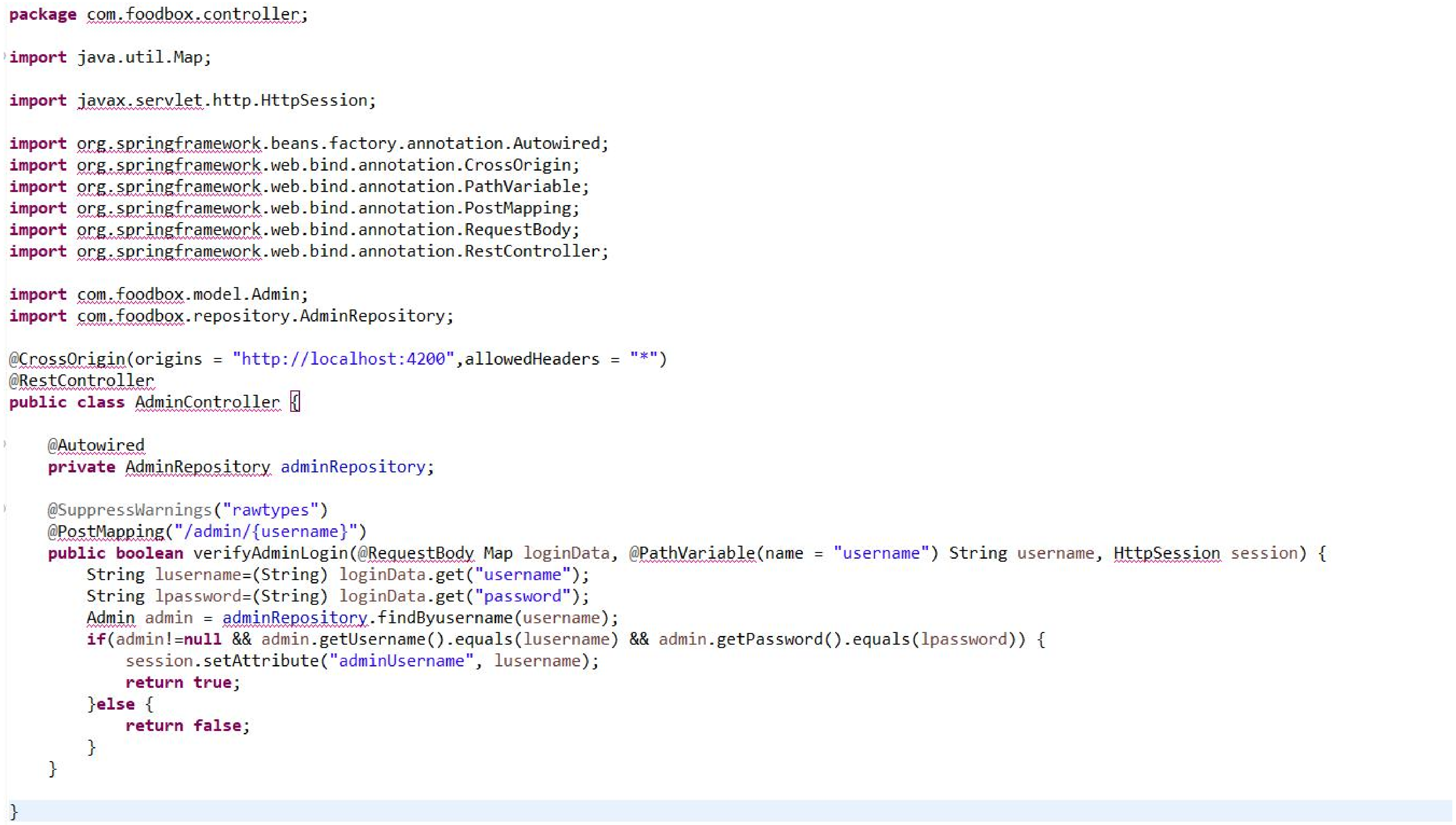
```java
private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";

public static void main(String args[]) {
    System.out.println("- Downloader started");
    File baseDirectory = new File(args[0]);
    System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());

    // If the maven-wrapper.properties exists, read it and check if it contains a custom
    // wrapperUrl parameter.
    File mavenWrapperPropertyFile = new File(baseDirectory, MAVEN_WRAPPER_PROPERTIES_PATH);
    String url = DEFAULT_DOWNLOAD_URL;
    if(mavenWrapperPropertyFile.exists()) {
        FileInputStream mavenWrapperPropertyFileInputStream = null;
        try {
            mavenWrapperPropertyFileInputStream = new FileInputStream(mavenWrapperPropertyFile);
            Properties mavenWrapperProperties = new Properties();
            mavenWrapperProperties.load(mavenWrapperPropertyFileInputStream);
            url = mavenWrapperProperties.getProperty(PROPERTY_NAME_WRAPPER_URL, url);
        } catch (IOException e) {
            System.out.println("- ERROR loading '" + MAVEN_WRAPPER_PROPERTIES_PATH + "'");
        } finally {
            try {
                if(mavenWrapperPropertyFileInputStream != null) {
                    mavenWrapperPropertyFileInputStream.close();
                }
            } catch (IOException e) {
                // Ignore ...
            }
        }
    }
    System.out.println("- Downloading from: " + url);

    File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
    if(!outputFile.getParentFile().exists()) {
        if(!outputFile.getParentFile().mkdirs()) {
            System.out.println(
                    "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
        }
    }
    System.out.println("- Downloading to: " + outputFile.getAbsolutePath());
    try {
        downloadFileFromURL(url, outputFile);
        System.out.println("Done");
```



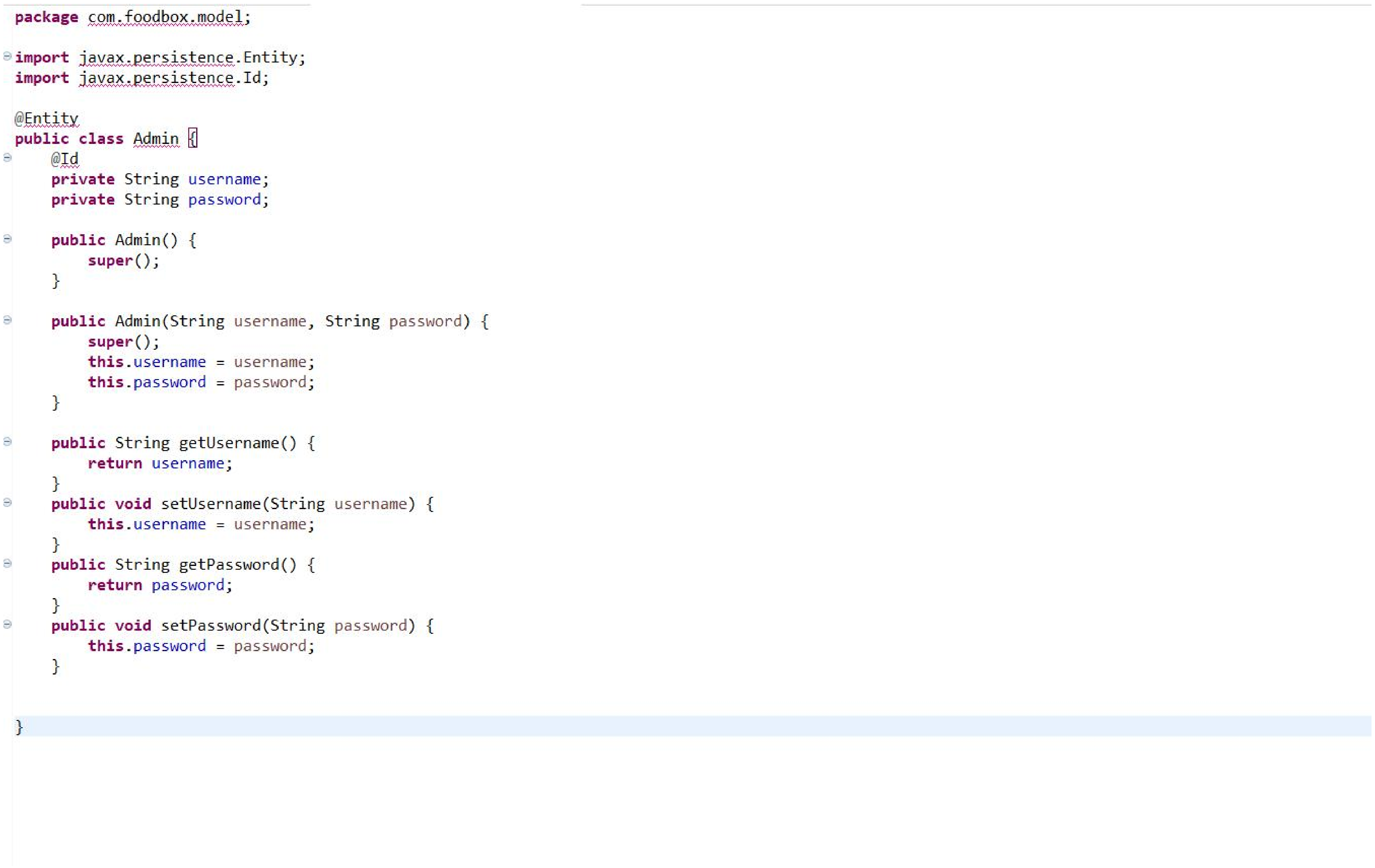
```
            }
        }
        System.out.println("- Downloading from: " + url);

        File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists()) {
            if(!outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");
            }
        }
        System.out.println("- Downloading to: " + outputFile.getAbsolutePath());
        try {
            downloadFileFromURL(url, outputFile);
            System.out.println("Done");
            System.exit(0);
        } catch (Throwable e) {
            System.out.println("- Error downloading");
            e.printStackTrace();
            System.exit(1);
        }
    }

    private static void downloadFileFromURL(String urlString, File destination) throws Exception {
        if (System.getenv("MVNW_USERNAME") != null && System.getenv("MVNW_PASSWORD") != null) {
            String username = System.getenv("MVNW_USERNAME");
            char[] password = System.getenv("MVNW_PASSWORD").toCharArray();
            Authenticator.setDefault(new Authenticator() {
                @Override
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(username, password);
                }
            });
        }
        URL website = new URL(urlString);
        ReadableByteChannel rbc;
        rbc = Channels.newChannel(website.openStream());
        FileOutputStream fos = new FileOutputStream(destination);
        fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
        fos.close();
        rbc.close();
    }

}
```



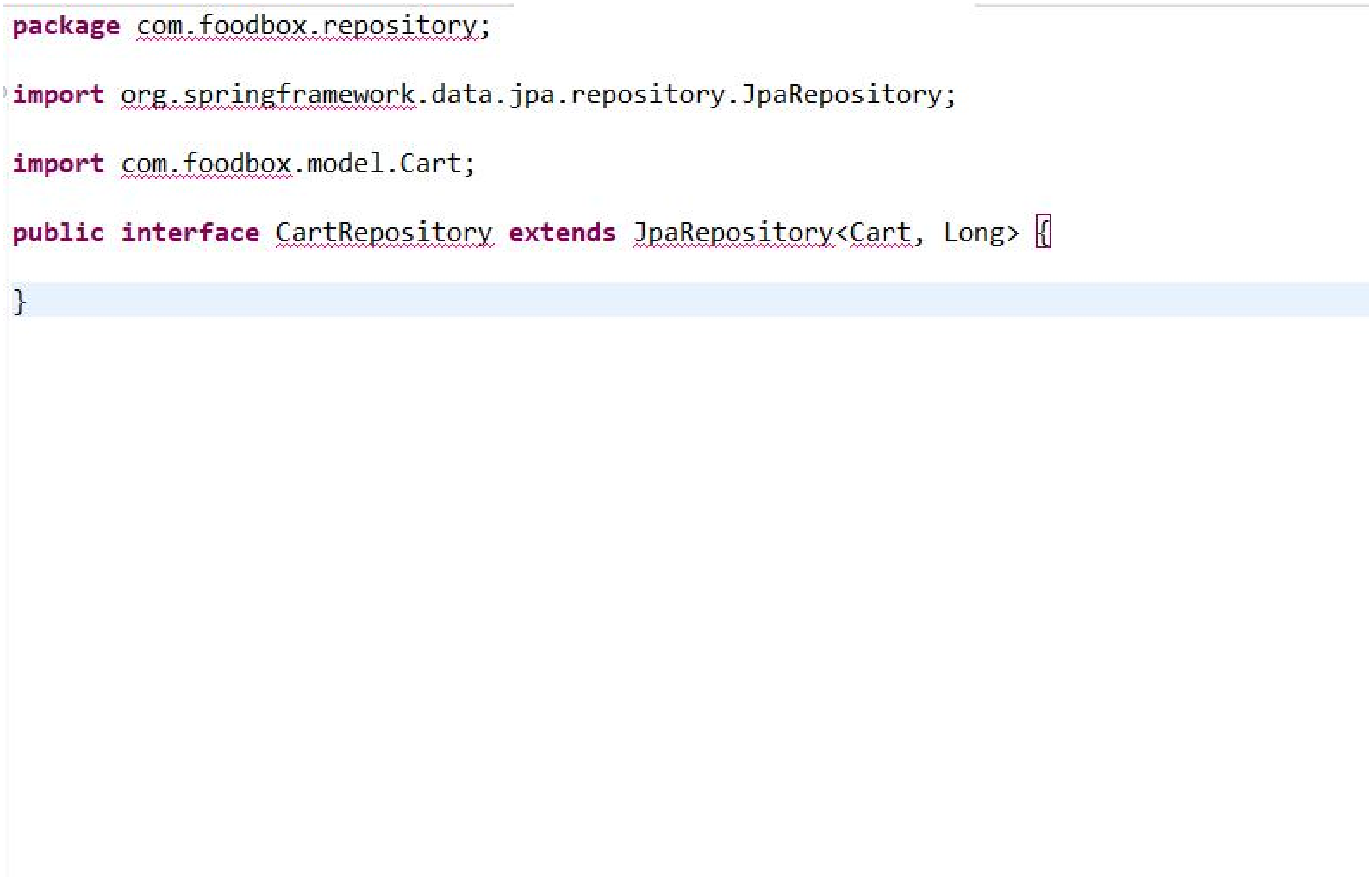
```java
package com.foodbox.config;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

@Component
@Order(Ordered.HIGHEST_PRECEDENCE)
public class ConfigCtrl implements Filter {
    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {
        final HttpServletResponse response = (HttpServletResponse) res;
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");
        response.setHeader("Access-Control-Allow-Headers", "Authorization, Content-Type");
        response.setHeader("Access-Control-Max-Age", "3600");
        if ("OPTIONS".equalsIgnoreCase(((HttpServletRequest) req).getMethod())) {
            response.setStatus(HttpServletResponse.SC_OK);
        } else {
            chain.doFilter(req, res);
        }
    }
    @Override
        public void destroy() {
    }
    @Override
        public void init(FilterConfig config) throws ServletException {
    }
}
```



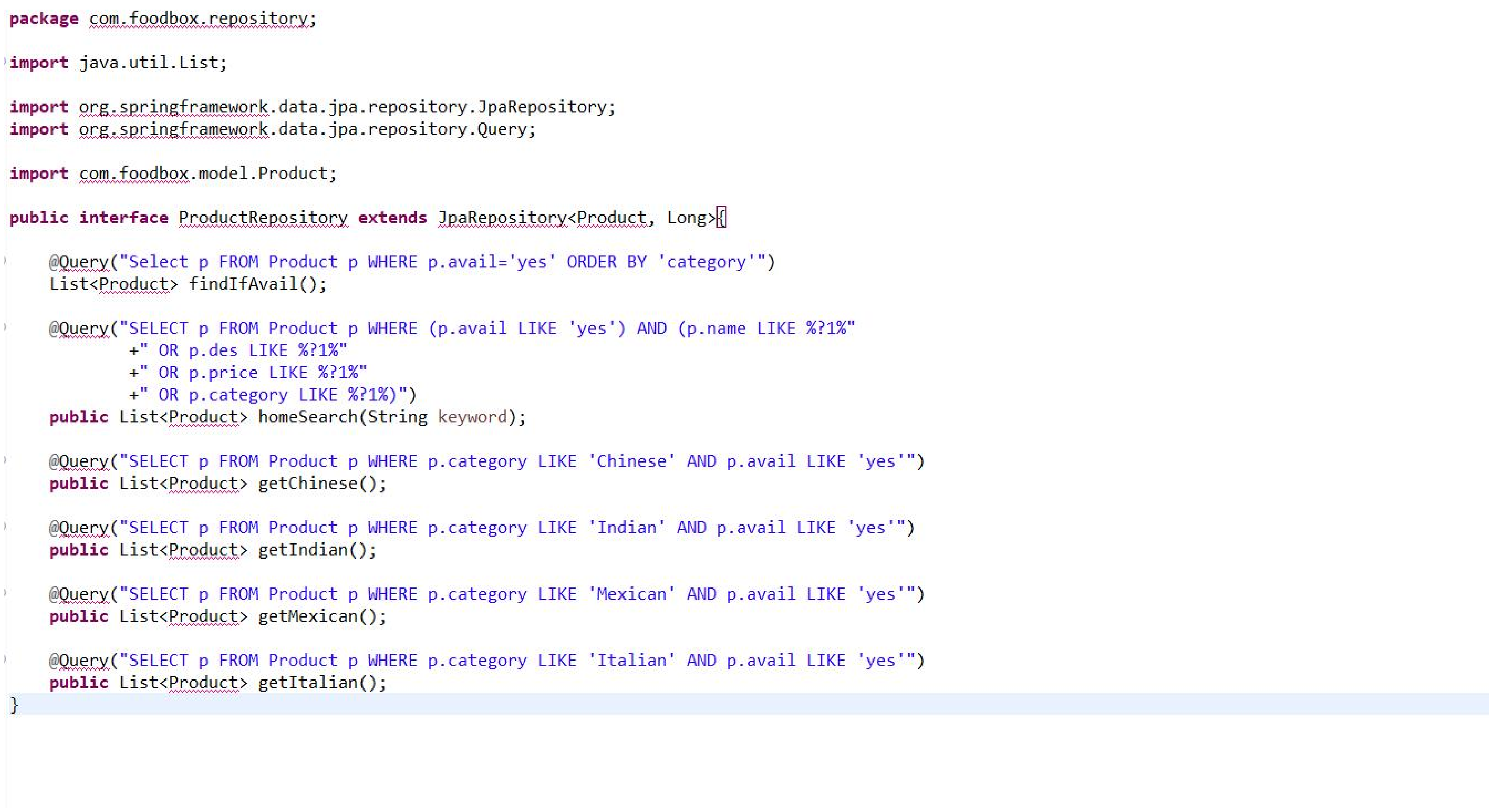
```java
package com.foodbox.controller;

import java.util.Map;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Admin;
import com.foodbox.repository.AdminRepository;

@CrossOrigin(origins = "http://localhost:4200",allowedHeaders = "*")
@RestController
public class AdminController {

    @Autowired
    private AdminRepository adminRepository;

    @SuppressWarnings("rawtypes")
    @PostMapping("/admin/{username}")
    public boolean verifyAdminLogin(@RequestBody Map loginData, @PathVariable(name = "username") String username, HttpSession session) {
        String lusername=(String) loginData.get("username");
        String lpassword=(String) loginData.get("password");
        Admin admin = adminRepository.findByusername(username);
        if(admin!=null && admin.getUsername().equals(lusername) && admin.getPassword().equals(lpassword)) {
            session.setAttribute("adminUsername", lusername);
            return true;
        }else {
            return false;
        }
    }

}
```



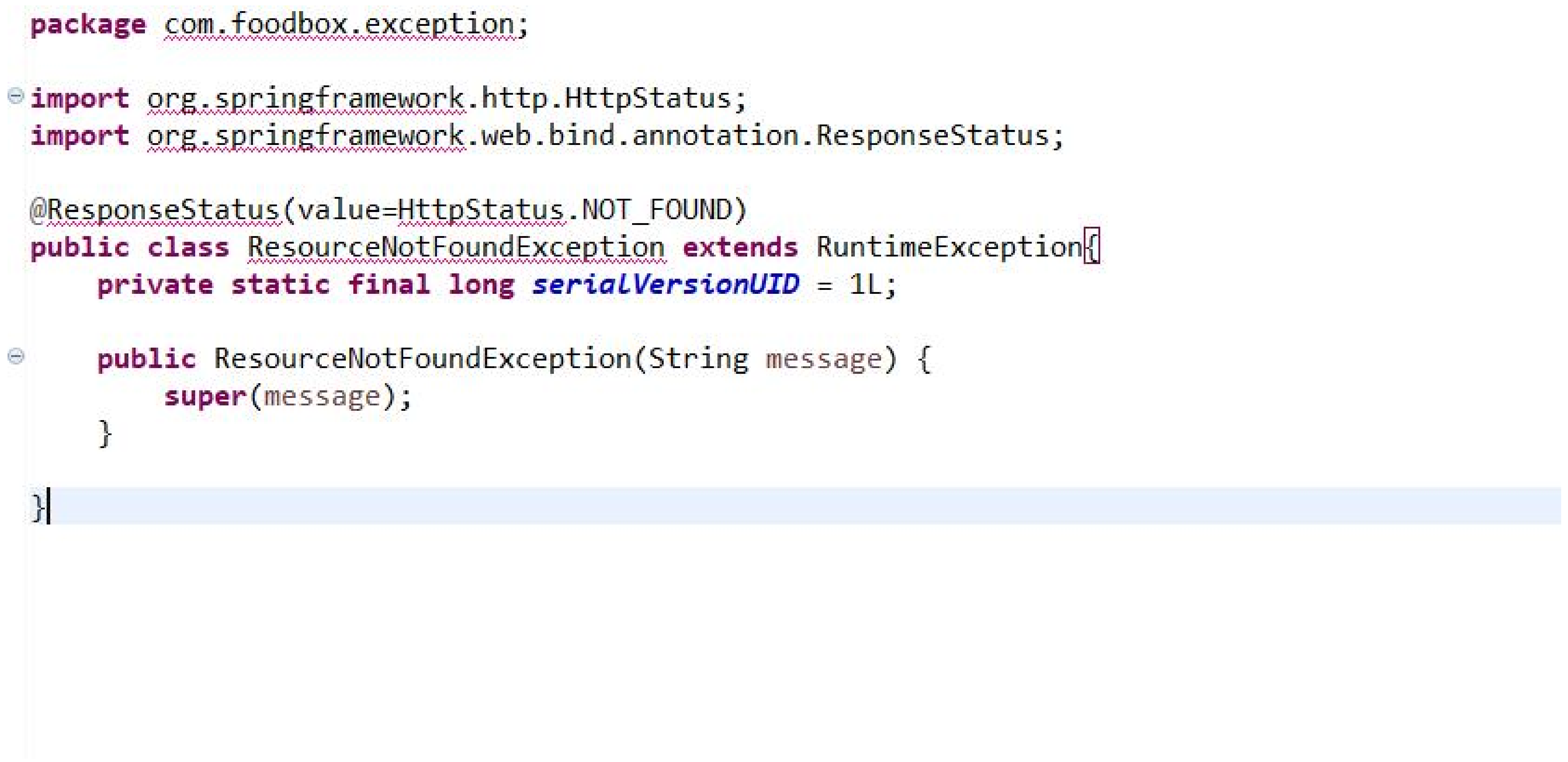
```java
package com.foodbox.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Admin {
    @Id
    private String username;
    private String password;

    public Admin() {
        super();
    }

    public Admin(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }


}
```



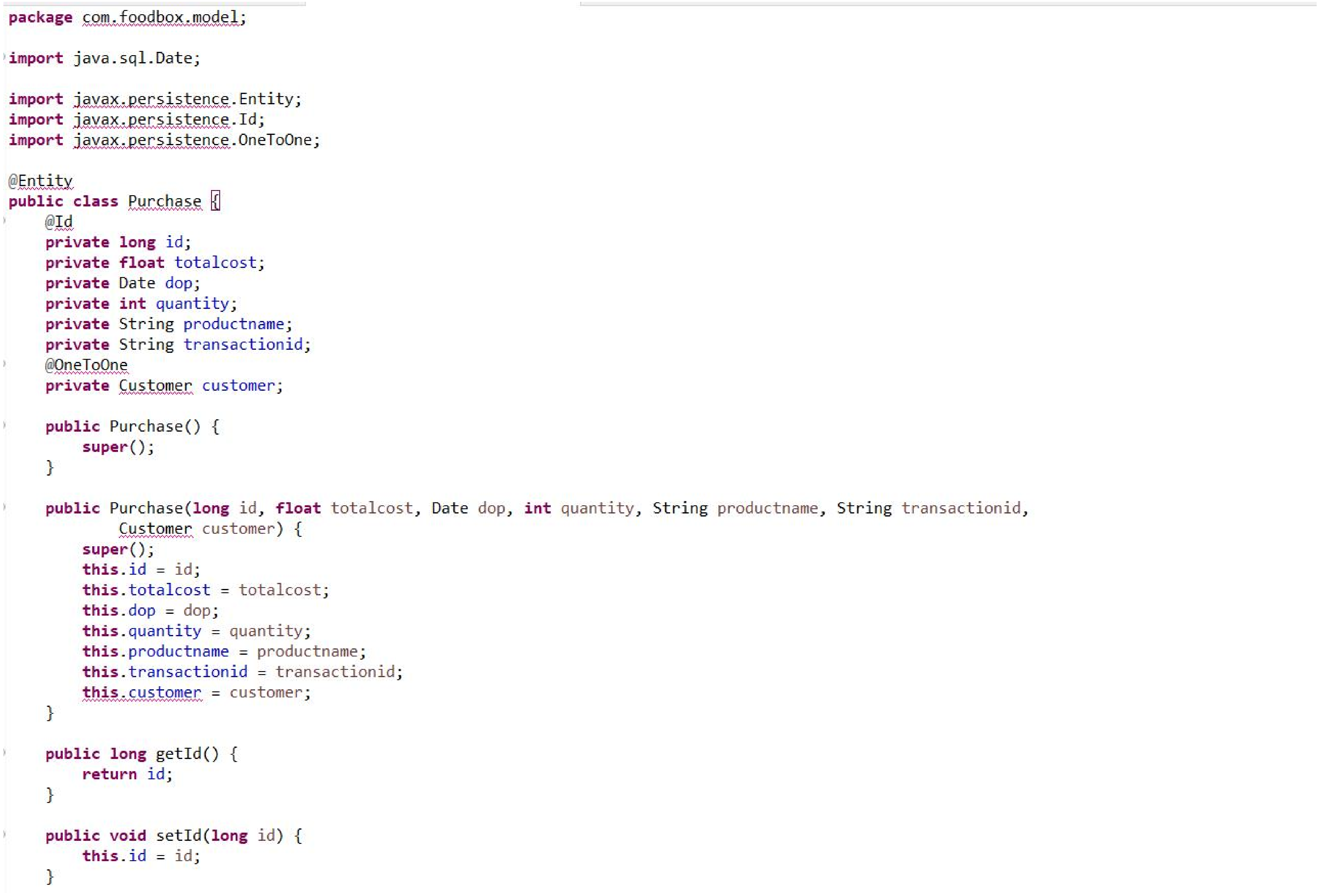
```
package com.foodbox.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.foodbox.model.Cart;

public interface CartRepository extends JpaRepository<Cart, Long> {

}
```



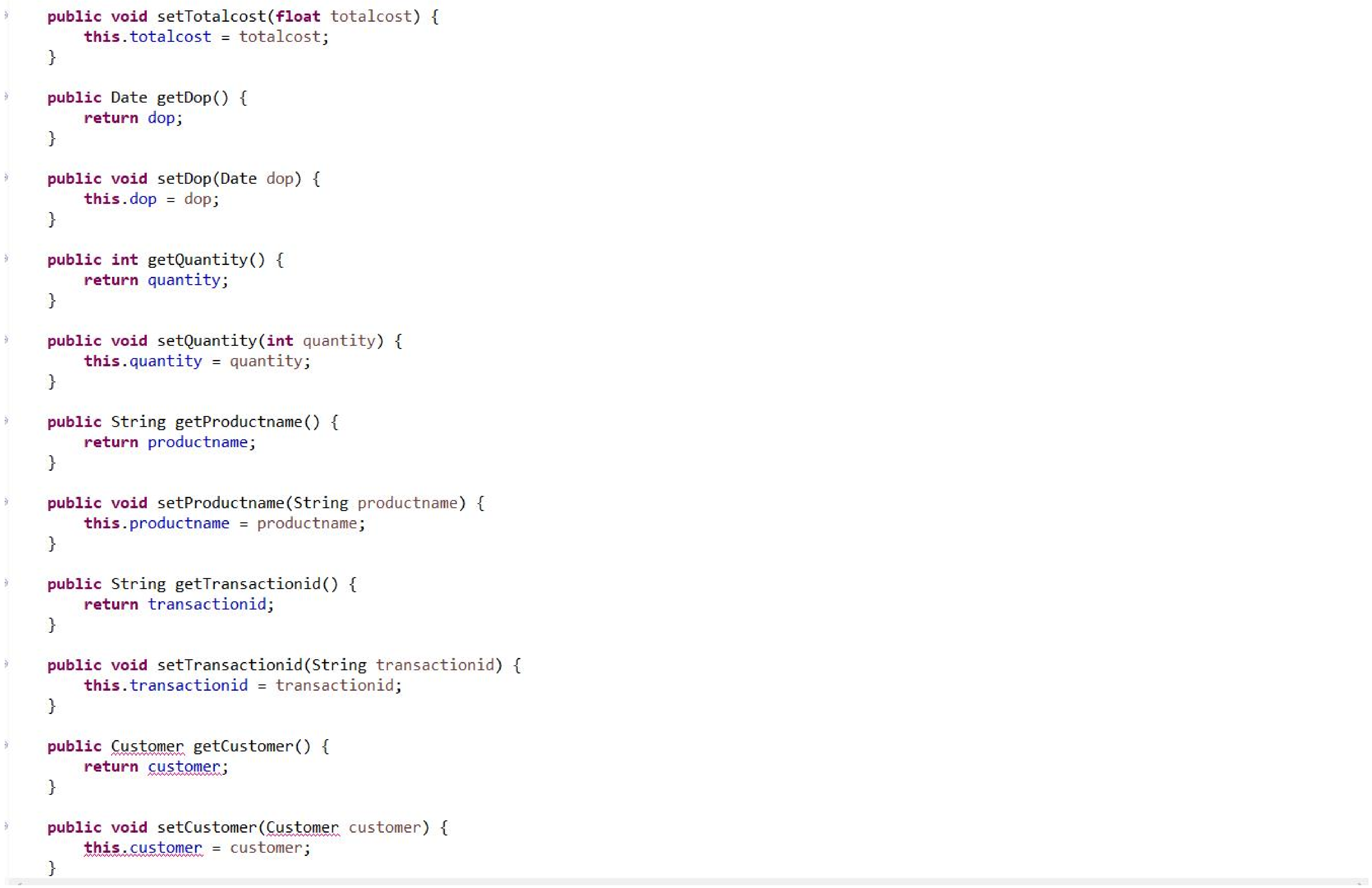
```java
package com.foodbox.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.foodbox.model.Product;

public interface ProductRepository extends JpaRepository<Product, Long>{

    @Query("Select p FROM Product p WHERE p.avail='yes' ORDER BY 'category'")
    List<Product> findIfAvail();

    @Query("SELECT p FROM Product p WHERE (p.avail LIKE 'yes') AND (p.name LIKE %?1%"
            +" OR p.des LIKE %?1%"
            +" OR p.price LIKE %?1%"
            +" OR p.category LIKE %?1%)")
    public List<Product> homeSearch(String keyword);

    @Query("SELECT p FROM Product p WHERE p.category LIKE 'Chinese' AND p.avail LIKE 'yes'")
    public List<Product> getChinese();

    @Query("SELECT p FROM Product p WHERE p.category LIKE 'Indian' AND p.avail LIKE 'yes'")
    public List<Product> getIndian();

    @Query("SELECT p FROM Product p WHERE p.category LIKE 'Mexican' AND p.avail LIKE 'yes'")
    public List<Product> getMexican();

    @Query("SELECT p FROM Product p WHERE p.category LIKE 'Italian' AND p.avail LIKE 'yes'")
    public List<Product> getItalian();
}
```



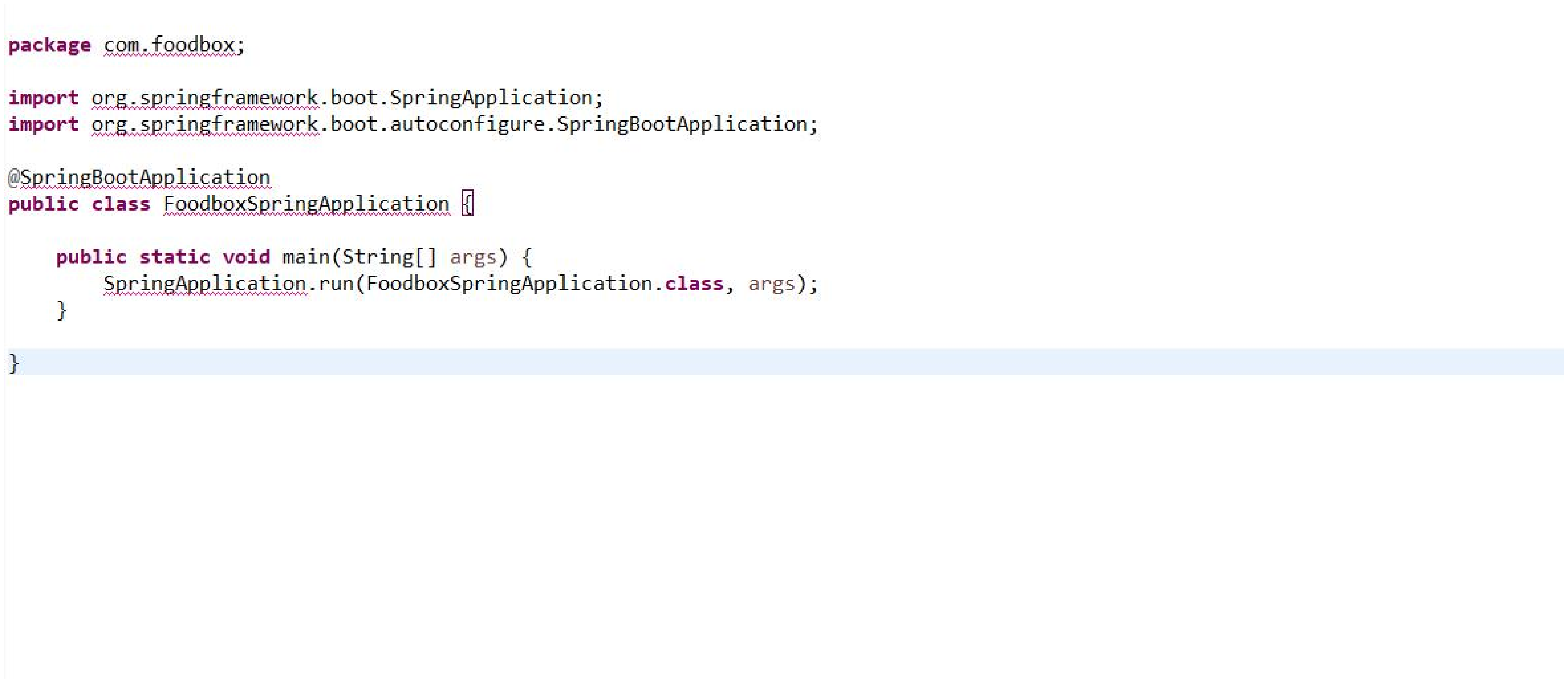
```java
package com.foodbox.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value=HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException{
    private static final long serialVersionUID = 1L;

    public ResourceNotFoundException(String message) {
        super(message);
    }

}
```



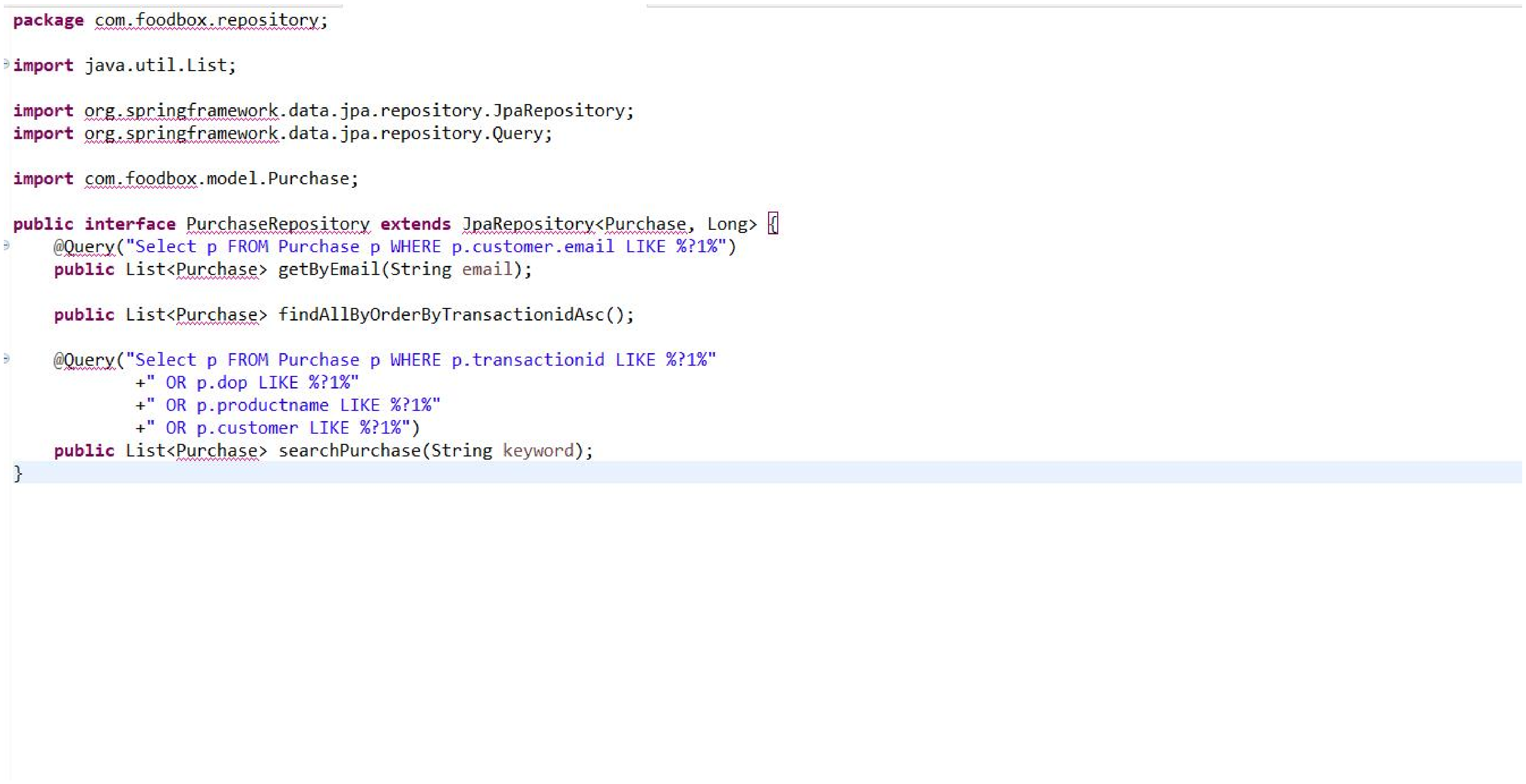
```java
package com.foodbox.model;

import java.sql.Date;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToOne;

@Entity
public class Purchase {
    @Id
    private long id;
    private float totalcost;
    private Date dop;
    private int quantity;
    private String productname;
    private String transactionid;
    @OneToOne
    private Customer customer;

    public Purchase() {
        super();
    }

    public Purchase(long id, float totalcost, Date dop, int quantity, String productname, String transactionid,
            Customer customer) {
        super();
        this.id = id;
        this.totalcost = totalcost;
        this.dop = dop;
        this.quantity = quantity;
        this.productname = productname;
        this.transactionid = transactionid;
        this.customer = customer;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
```



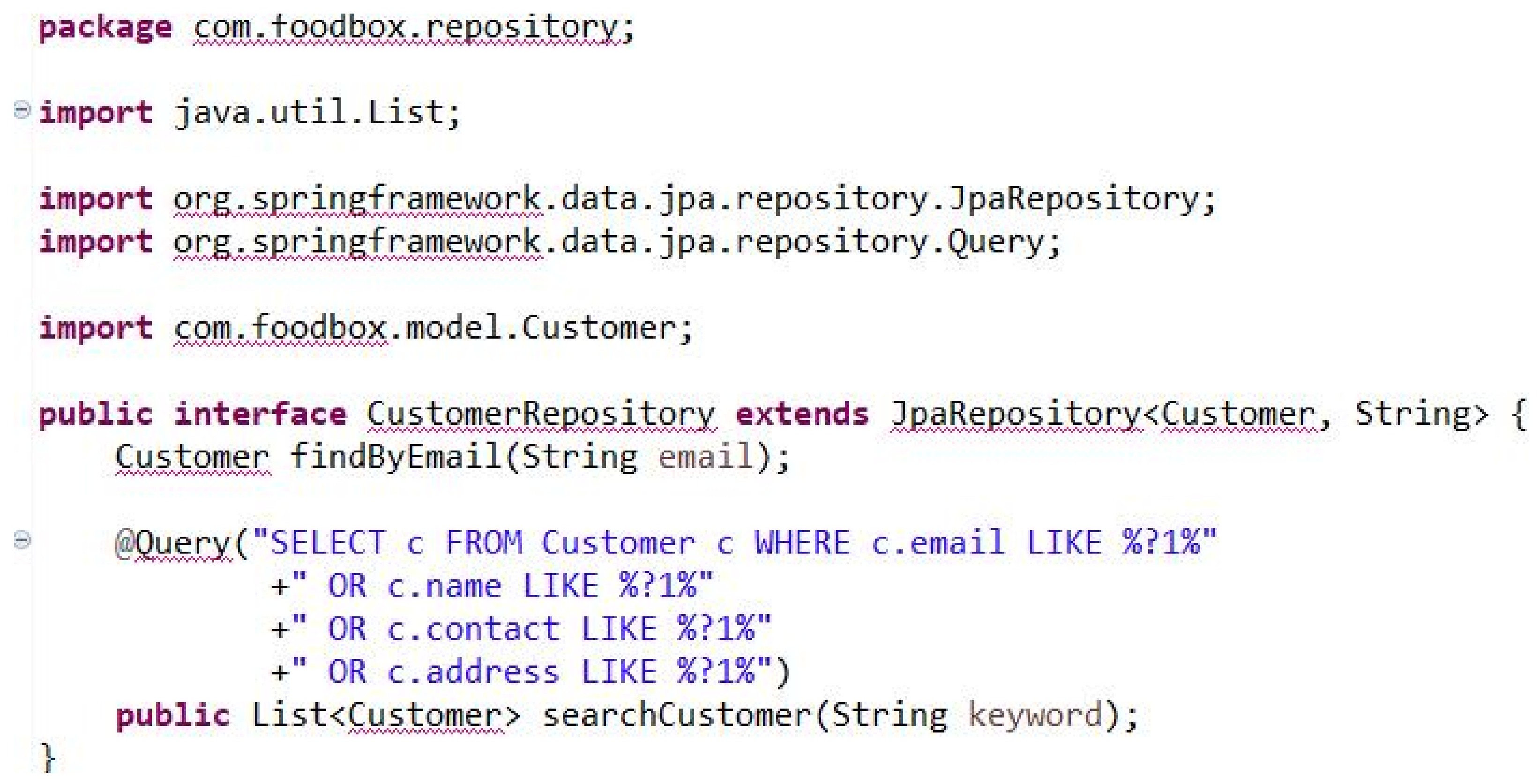
```
    public void setTotalcost(float totalcost) {
        this.totalcost = totalcost;
    }

    public Date getDop() {
        return dop;
    }

    public void setDop(Date dop) {
        this.dop = dop;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getProductname() {
        return productname;
    }

    public void setProductname(String productname) {
        this.productname = productname;
    }

    public String getTransactionid() {
        return transactionid;
    }

    public void setTransactionid(String transactionid) {
        this.transactionid = transactionid;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }
```



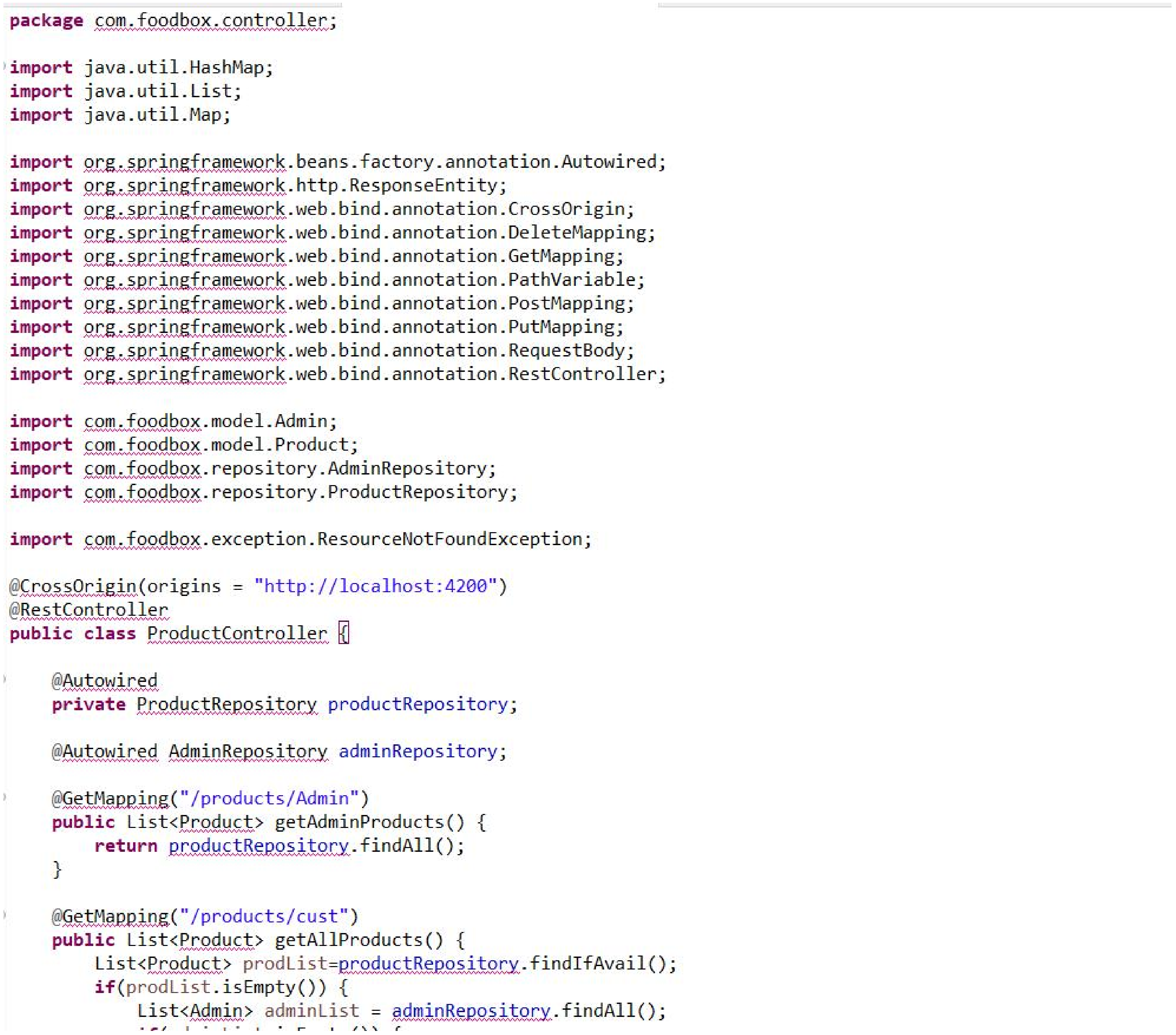
```java
package com.foodbox;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FoodboxSpringApplication {

    public static void main(String[] args) {
        SpringApplication.run(FoodboxSpringApplication.class, args);
    }

}
```



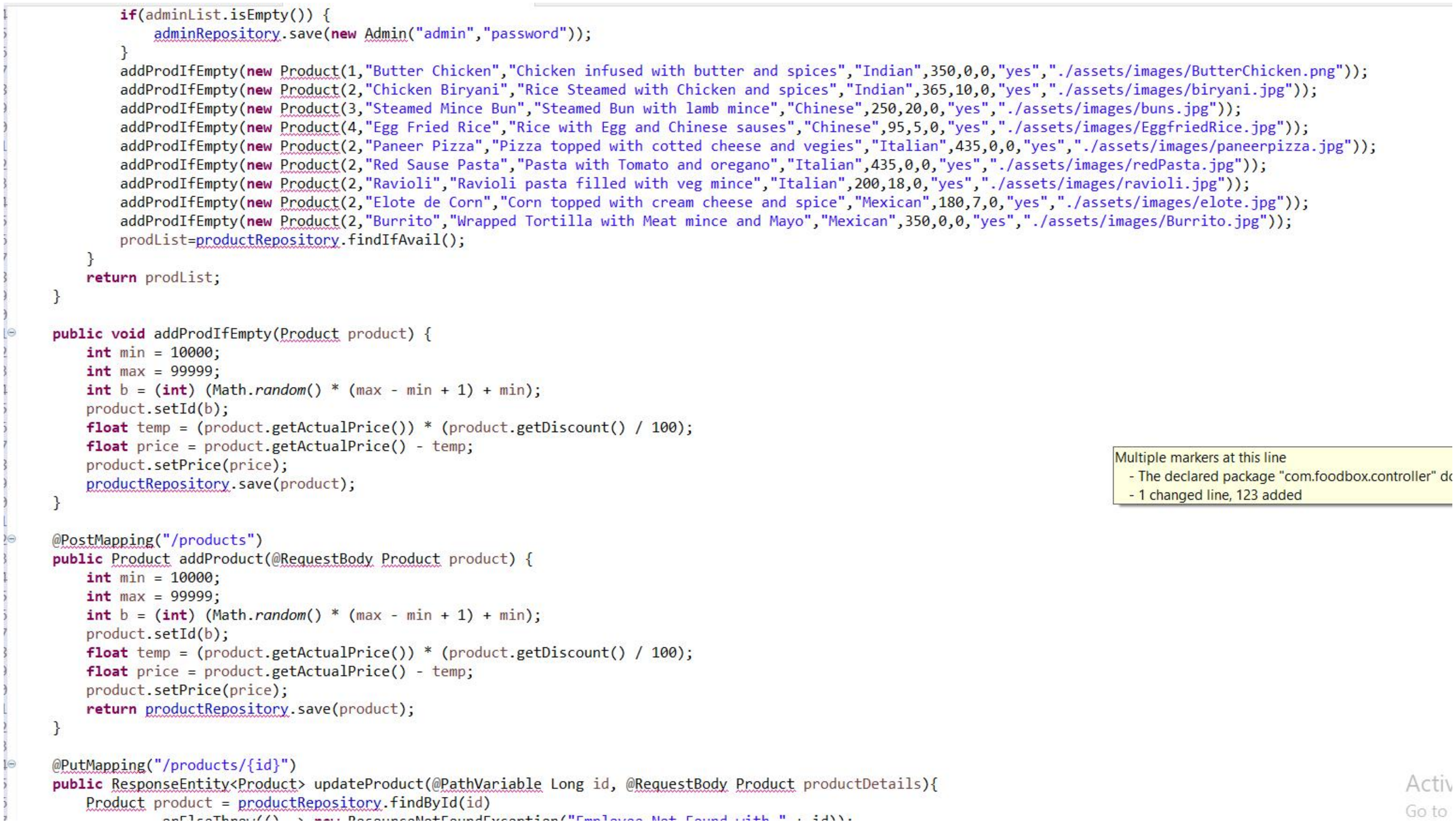
```java
package com.foodbox.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.foodbox.model.Purchase;

public interface PurchaseRepository extends JpaRepository<Purchase, Long> {
    @Query("Select p FROM Purchase p WHERE p.customer.email LIKE %?1%")
    public List<Purchase> getByEmail(String email);

    public List<Purchase> findAllByOrderByTransactionidAsc();

    @Query("Select p FROM Purchase p WHERE p.transactionid LIKE %?1%"
            +" OR p.dop LIKE %?1%"
            +" OR p.productname LIKE %?1%"
            +" OR p.customer LIKE %?1%")
    public List<Purchase> searchPurchase(String keyword);
}
```



```java
package com.foodbox.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.foodbox.model.Customer;

public interface CustomerRepository extends JpaRepository<Customer, String> {
    Customer findByEmail(String email);

    @Query("SELECT c FROM Customer c WHERE c.email LIKE %?1%"
            +" OR c.name LIKE %?1%"
            +" OR c.contact LIKE %?1%"
            +" OR c.address LIKE %?1%")
    public List<Customer> searchCustomer(String keyword);
}
```

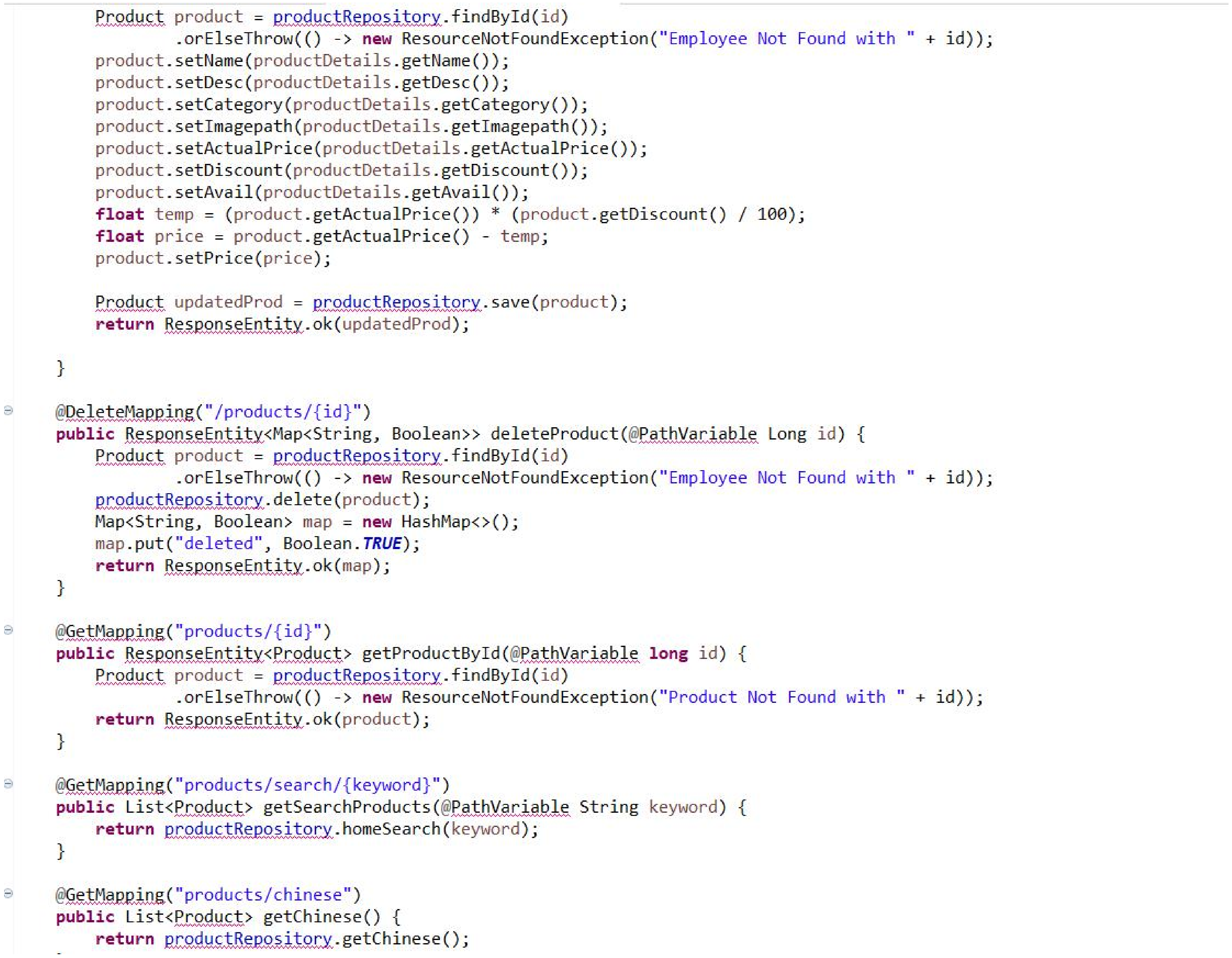


```java
package com.foodbox.controller;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.foodbox.model.Admin;
import com.foodbox.model.Product;
import com.foodbox.repository.AdminRepository;
import com.foodbox.repository.ProductRepository;

import com.foodbox.exception.ResourceNotFoundException;

@CrossOrigin(origins = "http://localhost:4200")
@RestController
public class ProductController {

    @Autowired
    private ProductRepository productRepository;

    @Autowired AdminRepository adminRepository;

    @GetMapping("/products/Admin")
    public List<Product> getAdminProducts() {
        return productRepository.findAll();
    }

    @GetMapping("/products/cust")
    public List<Product> getAllProducts() {
        List<Product> prodList=productRepository.findIfAvail();
        if(prodList.isEmpty()) {
            List<Admin> adminList = adminRepository.findAll();
```

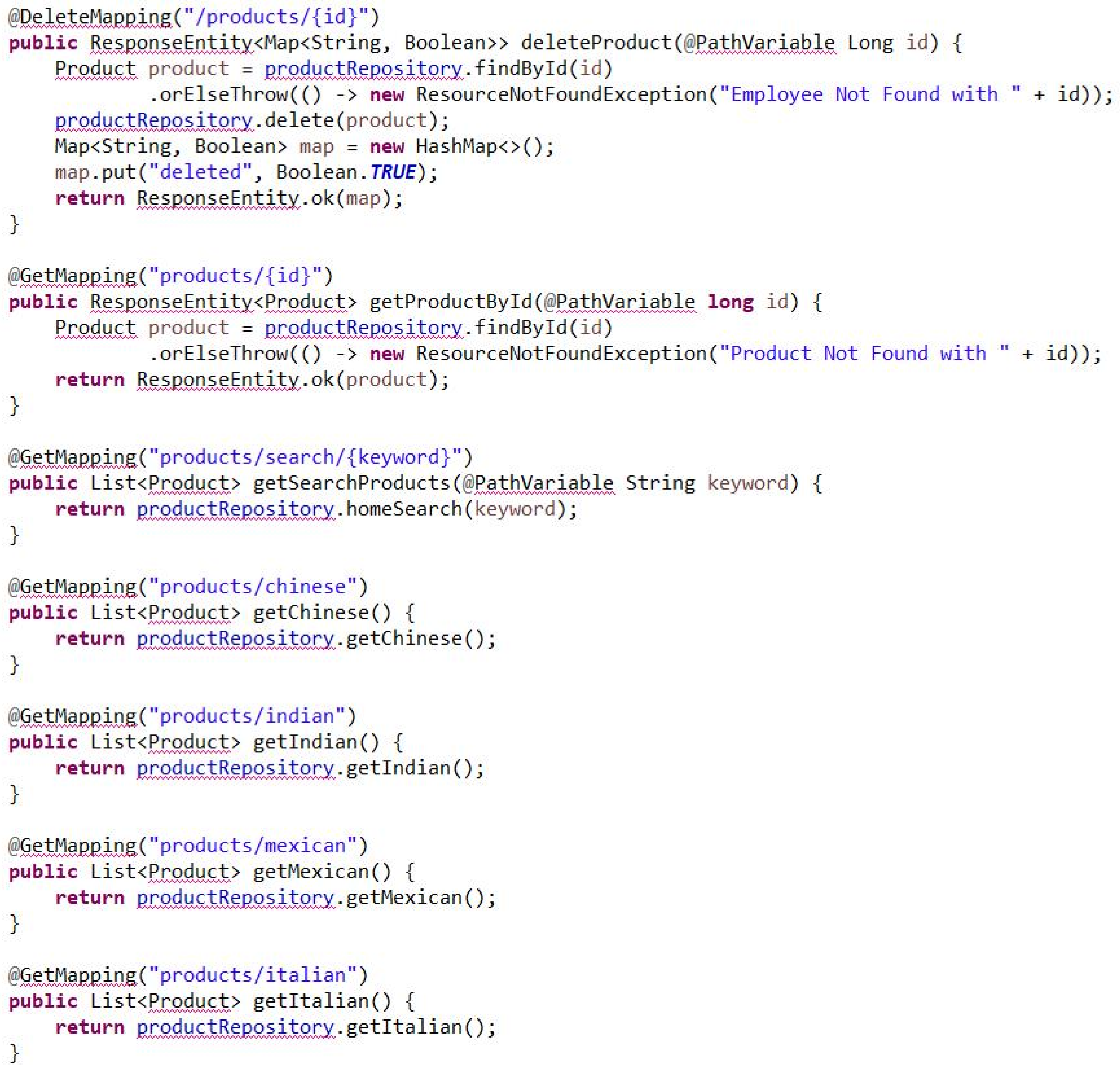


```java
            if(adminList.isEmpty()) {
                adminRepository.save(new Admin("admin","password"));
            }
            addProdIfEmpty(new Product(1,"Butter Chicken","Chicken infused with butter and spices","Indian",350,0,0,"yes","./assets/images/ButterChicken.png"));
            addProdIfEmpty(new Product(2,"Chicken Biryani","Rice Steamed with Chicken and spices","Indian",365,10,0,"yes","./assets/images/biryani.jpg"));
            addProdIfEmpty(new Product(3,"Steamed Mince Bun","Steamed Bun with lamb mince","Chinese",250,20,0,"yes","./assets/images/buns.jpg"));
            addProdIfEmpty(new Product(4,"Egg Fried Rice","Rice with Egg and Chinese sauses","Chinese",95,5,0,"yes","./assets/images/EggfriedRice.jpg"));
            addProdIfEmpty(new Product(2,"Paneer Pizza","Pizza topped with cotted cheese and vegies","Italian",435,0,0,"yes","./assets/images/paneerpizza.jpg"));
            addProdIfEmpty(new Product(2,"Red Sause Pasta","Pasta with Tomato and oregano","Italian",435,0,0,"yes","./assets/images/redPasta.jpg"));
            addProdIfEmpty(new Product(2,"Ravioli","Ravioli pasta filled with veg mince","Italian",200,18,0,"yes","./assets/images/ravioli.jpg"));
            addProdIfEmpty(new Product(2,"Elote de Corn","Corn topped with cream cheese and spice","Mexican",180,7,0,"yes","./assets/images/elote.jpg"));
            addProdIfEmpty(new Product(2,"Burrito","Wrapped Tortilla with Meat mince and Mayo","Mexican",350,0,0,"yes","./assets/images/Burrito.jpg"));
            prodList=productRepository.findIfAvail();
        }
        return prodList;
    }

    public void addProdIfEmpty(Product product) {
        int min = 10000;
        int max = 99999;
        int b = (int) (Math.random() * (max - min + 1) + min);
        product.setId(b);
        float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
        float price = product.getActualPrice() - temp;
        product.setPrice(price);
        productRepository.save(product);
    }

    @PostMapping("/products")
    public Product addProduct(@RequestBody Product product) {
        int min = 10000;
        int max = 99999;
        int b = (int) (Math.random() * (max - min + 1) + min);
        product.setId(b);
        float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
        float price = product.getActualPrice() - temp;
        product.setPrice(price);
        return productRepository.save(product);
    }

    @PutMapping("/products/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product productDetails){
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));
```

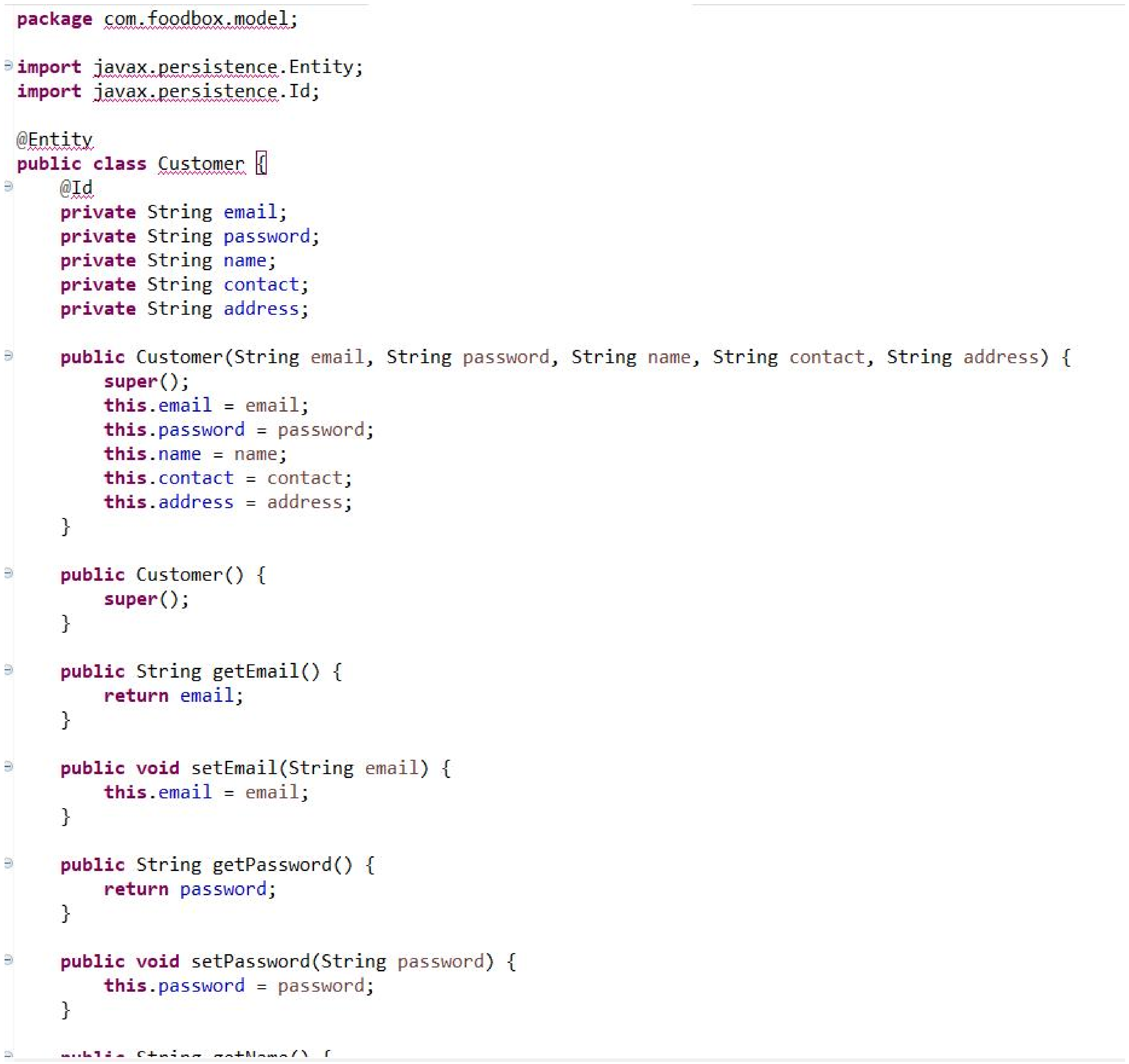
```
Multiple markers at this line
  - The declared package "com.foodbox.controller" do
  - 1 changed line, 123 added
```



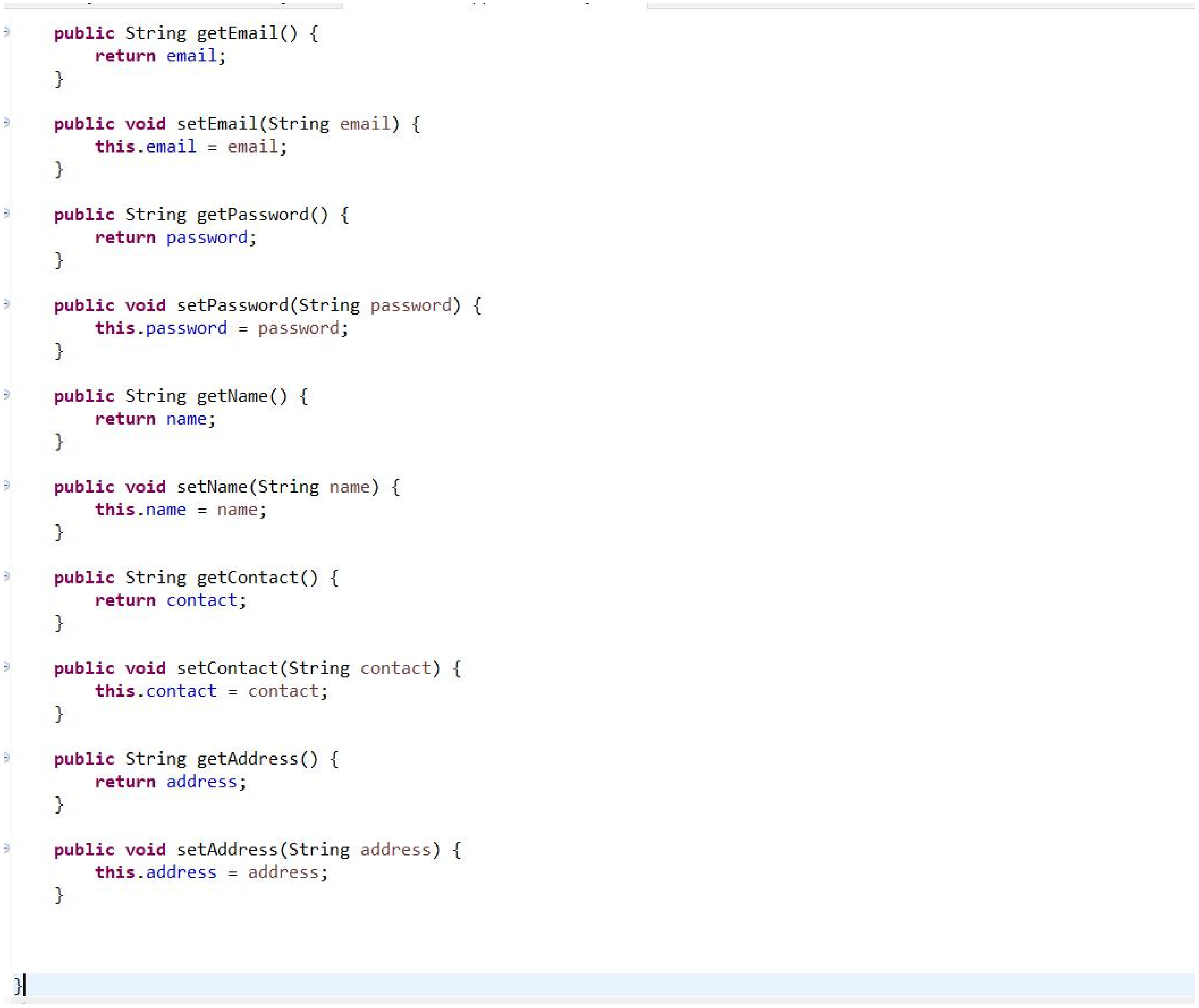
```java
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));
        product.setName(productDetails.getName());
        product.setDesc(productDetails.getDesc());
        product.setCategory(productDetails.getCategory());
        product.setImagepath(productDetails.getImagepath());
        product.setActualPrice(productDetails.getActualPrice());
        product.setDiscount(productDetails.getDiscount());
        product.setAvail(productDetails.getAvail());
        float temp = (product.getActualPrice()) * (product.getDiscount() / 100);
        float price = product.getActualPrice() - temp;
        product.setPrice(price);

        Product updatedProd = productRepository.save(product);
        return ResponseEntity.ok(updatedProd);

    }

    @DeleteMapping("/products/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id) {
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));
        productRepository.delete(product);
        Map<String, Boolean> map = new HashMap<>();
        map.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(map);
    }

    @GetMapping("products/{id}")
    public ResponseEntity<Product> getProductById(@PathVariable long id) {
        Product product = productRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Product Not Found with " + id));
        return ResponseEntity.ok(product);
    }

    @GetMapping("products/search/{keyword}")
    public List<Product> getSearchProducts(@PathVariable String keyword) {
        return productRepository.homeSearch(keyword);
    }

    @GetMapping("products/chinese")
    public List<Product> getChinese() {
        return productRepository.getChinese();
```



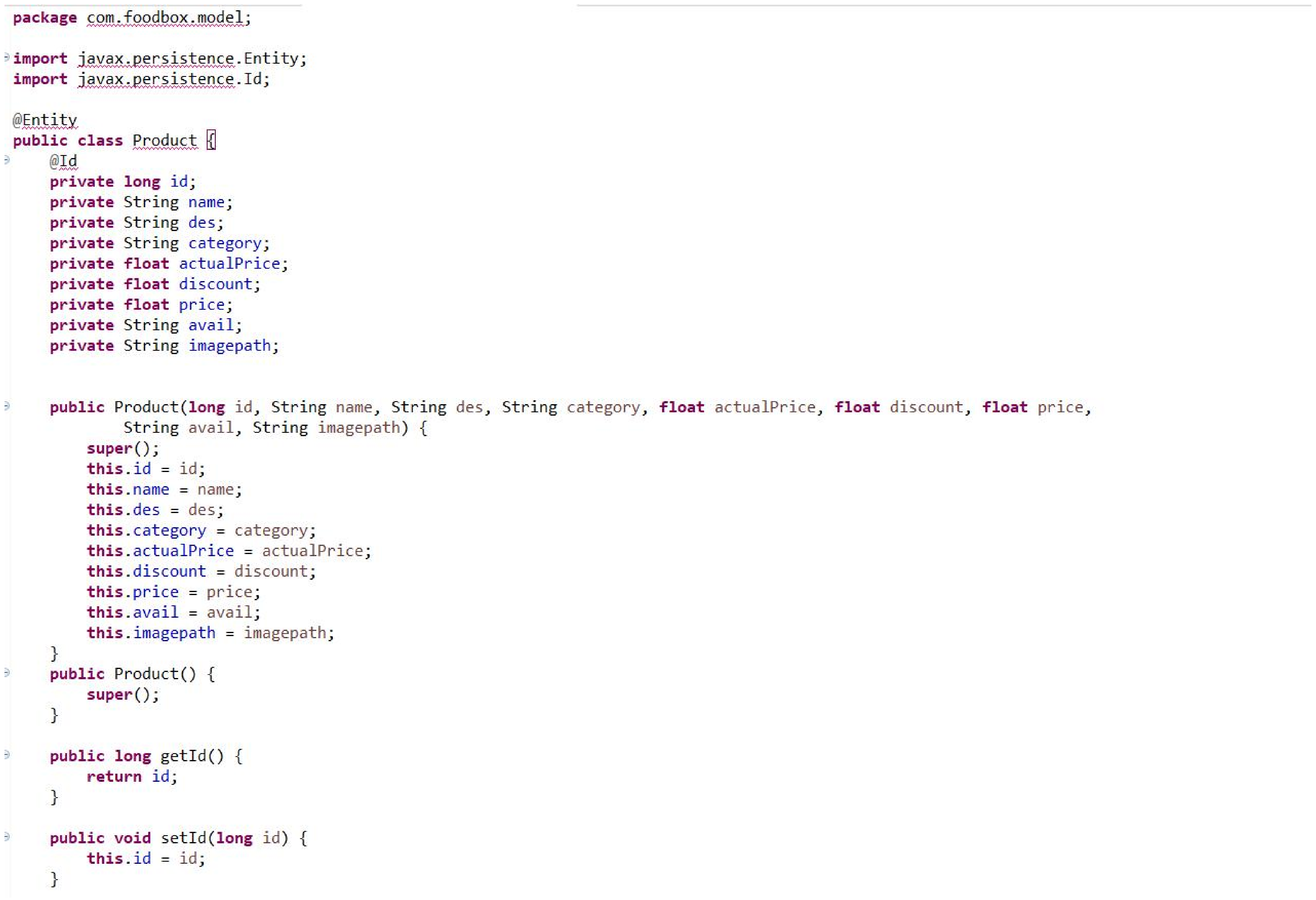
```java
@DeleteMapping("/products/{id}")
public ResponseEntity<Map<String, Boolean>> deleteProduct(@PathVariable Long id) {
    Product product = productRepository.findById(id)
            .orElseThrow(() -> new ResourceNotFoundException("Employee Not Found with " + id));
    productRepository.delete(product);
    Map<String, Boolean> map = new HashMap<>();
    map.put("deleted", Boolean.TRUE);
    return ResponseEntity.ok(map);
}

@GetMapping("products/{id}")
public ResponseEntity<Product> getProductById(@PathVariable long id) {
    Product product = productRepository.findById(id)
            .orElseThrow(() -> new ResourceNotFoundException("Product Not Found with " + id));
    return ResponseEntity.ok(product);
}

@GetMapping("products/search/{keyword}")
public List<Product> getSearchProducts(@PathVariable String keyword) {
    return productRepository.homeSearch(keyword);
}

@GetMapping("products/chinese")
public List<Product> getChinese() {
    return productRepository.getChinese();
}

@GetMapping("products/indian")
public List<Product> getIndian() {
    return productRepository.getIndian();
}

@GetMapping("products/mexican")
public List<Product> getMexican() {
    return productRepository.getMexican();
}

@GetMapping("products/italian")
public List<Product> getItalian() {
    return productRepository.getItalian();
}
```



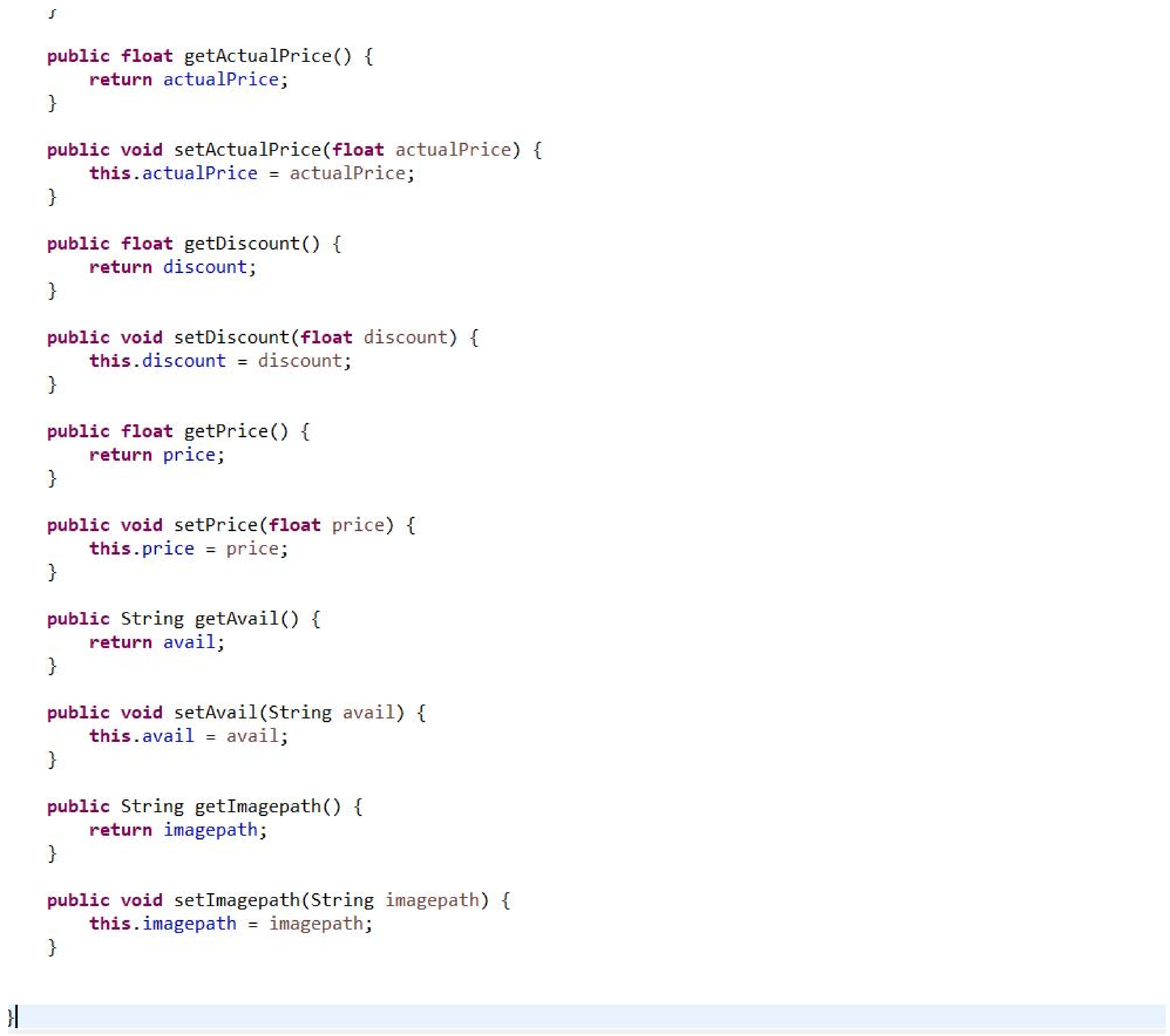
```java
package com.foodbox.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Customer {
    @Id
    private String email;
    private String password;
    private String name;
    private String contact;
    private String address;

    public Customer(String email, String password, String name, String contact, String address) {
        super();
        this.email = email;
        this.password = password;
        this.name = name;
        this.contact = contact;
        this.address = address;
    }

    public Customer() {
        super();
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getName() {
```



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

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getContact() {
        return contact;
    }

    public void setContact(String contact) {
        this.contact = contact;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }


}
```



```java
package com.foodbox.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Product {
    @Id
    private long id;
    private String name;
    private String des;
    private String category;
    private float actualPrice;
    private float discount;
    private float price;
    private String avail;
    private String imagepath;


    public Product(long id, String name, String des, String category, float actualPrice, float discount, float price,
            String avail, String imagepath) {
        super();
        this.id = id;
        this.name = name;
        this.des = des;
        this.category = category;
        this.actualPrice = actualPrice;
        this.discount = discount;
        this.price = price;
        this.avail = avail;
        this.imagepath = imagepath;
    }
    public Product() {
        super();
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDesc() {
        return des;
    }

    public void setDesc(String des) {
        this.des = des;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public float getActualPrice() {
        return actualPrice;
    }

    public void setActualPrice(float actualPrice) {
        this.actualPrice = actualPrice;
    }

    public float getDiscount() {
        return discount;
    }

    public void setDiscount(float discount) {
        this.discount = discount;
    }

    public float getPrice() {
        return price;
    }
```



```java
    }

    public float getActualPrice() {
        return actualPrice;
    }

    public void setActualPrice(float actualPrice) {
        this.actualPrice = actualPrice;
    }

    public float getDiscount() {
        return discount;
    }

    public void setDiscount(float discount) {
        this.discount = discount;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public String getAvail() {
        return avail;
    }

    public void setAvail(String avail) {
        this.avail = avail;
    }

    public String getImagepath() {
        return imagepath;
    }

    public void setImagepath(String imagepath) {
        this.imagepath = imagepath;
    }


}
```