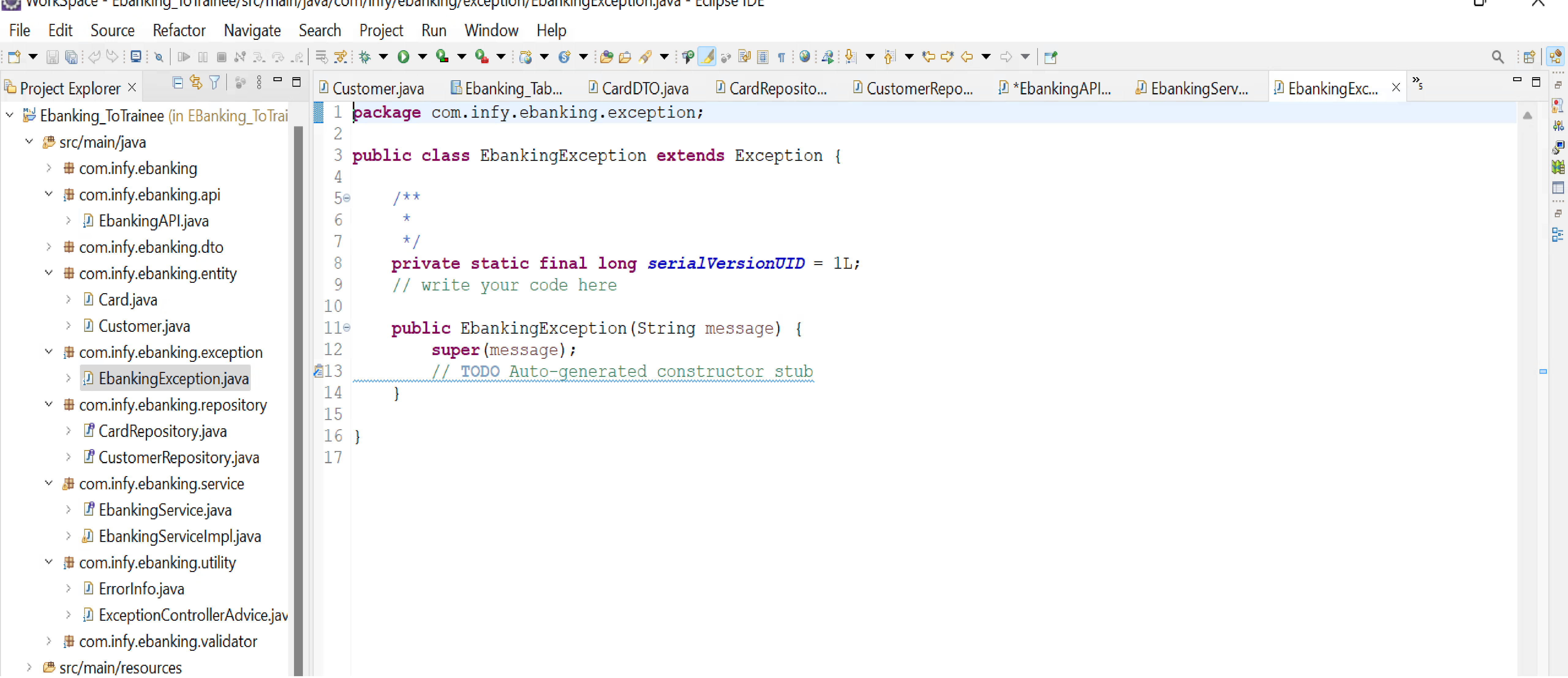
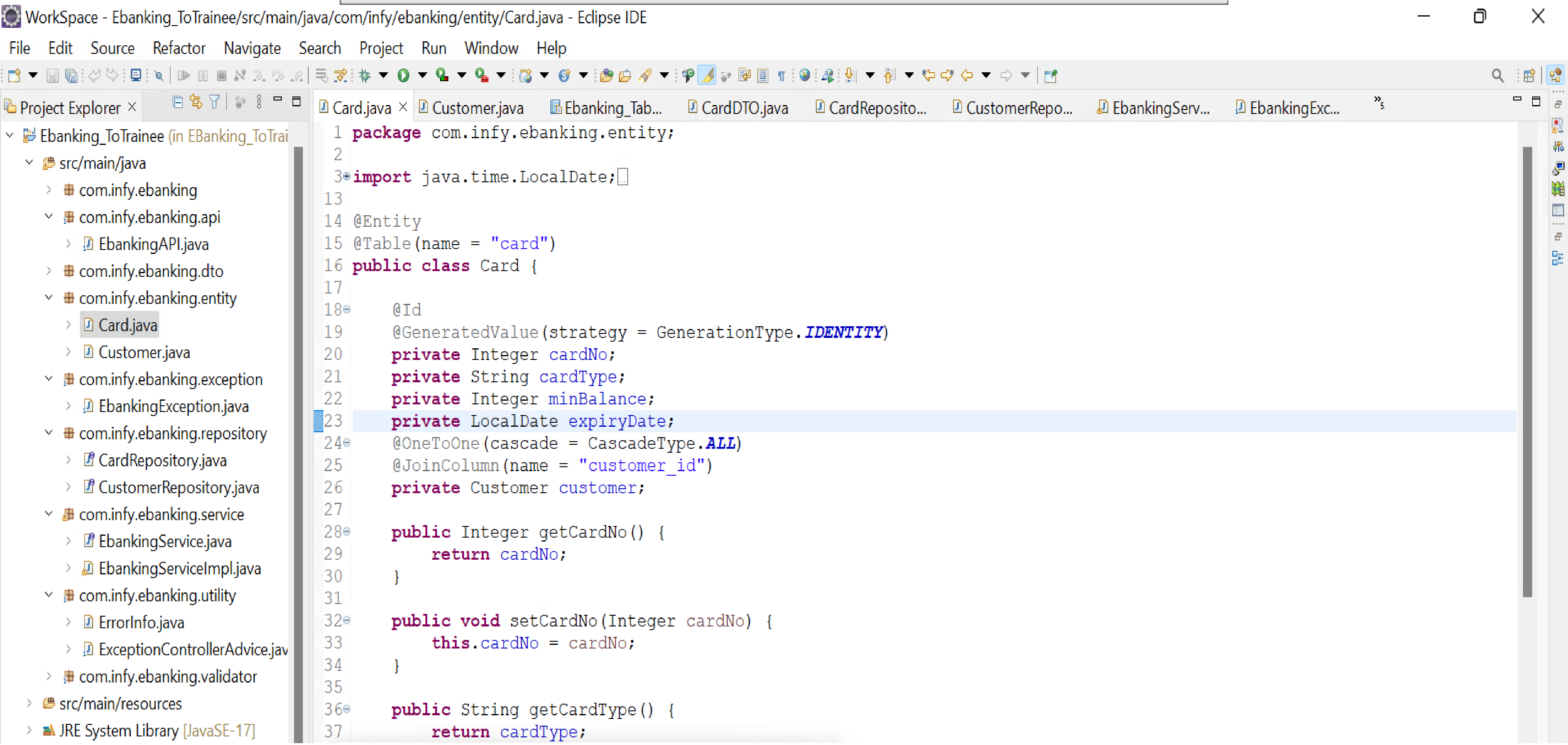
Step 1. write exception class

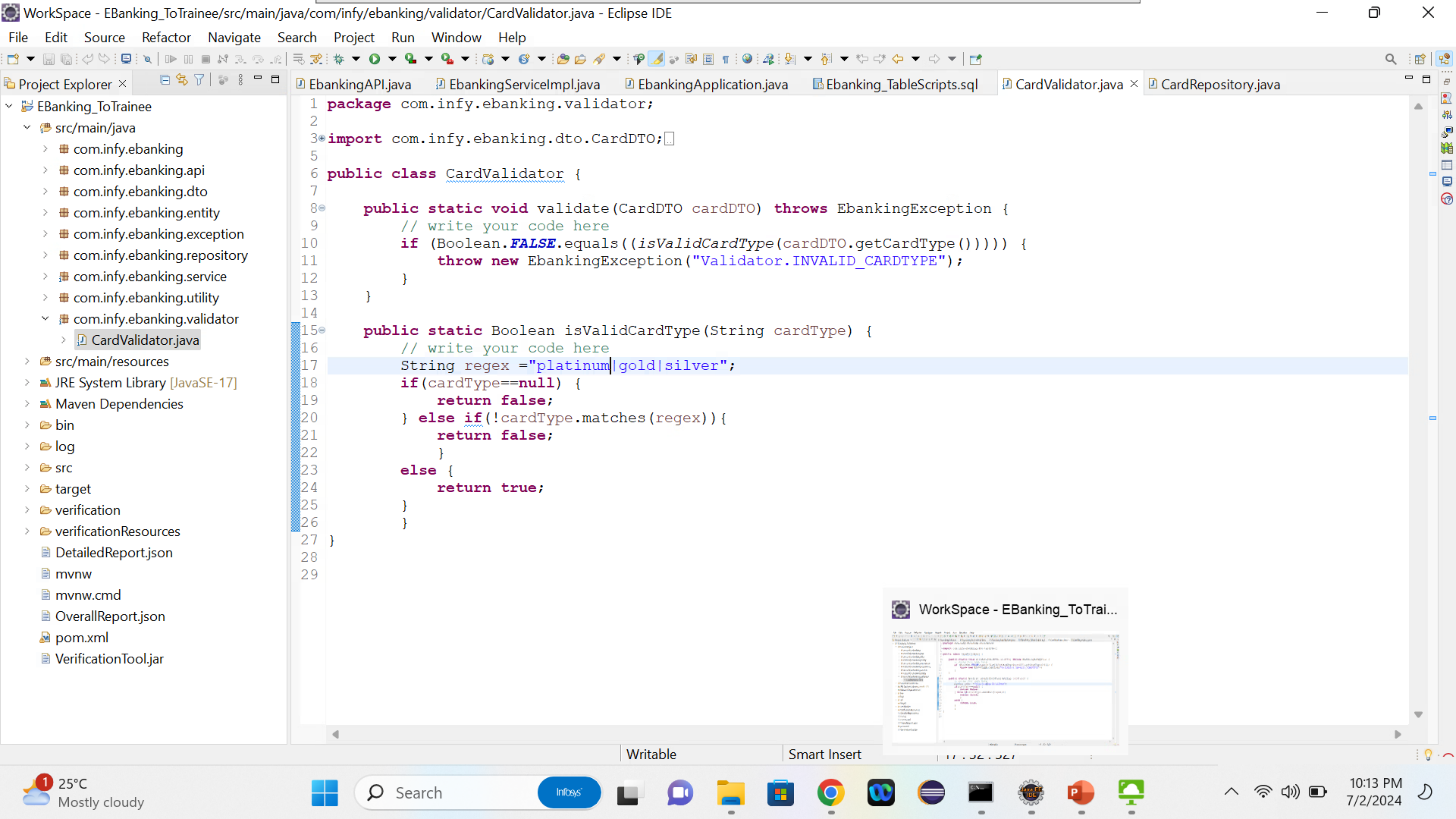


```java
package com.infy.ebanking.exception;

public class EbankingException extends Exception {

	/**
	 * 
	 */
	private static final long serialVersionUID = 1L;
	// write your code here

	public EbankingException(String message) {
		super(message);
		// TODO Auto-generated constructor stub
	}

}
```



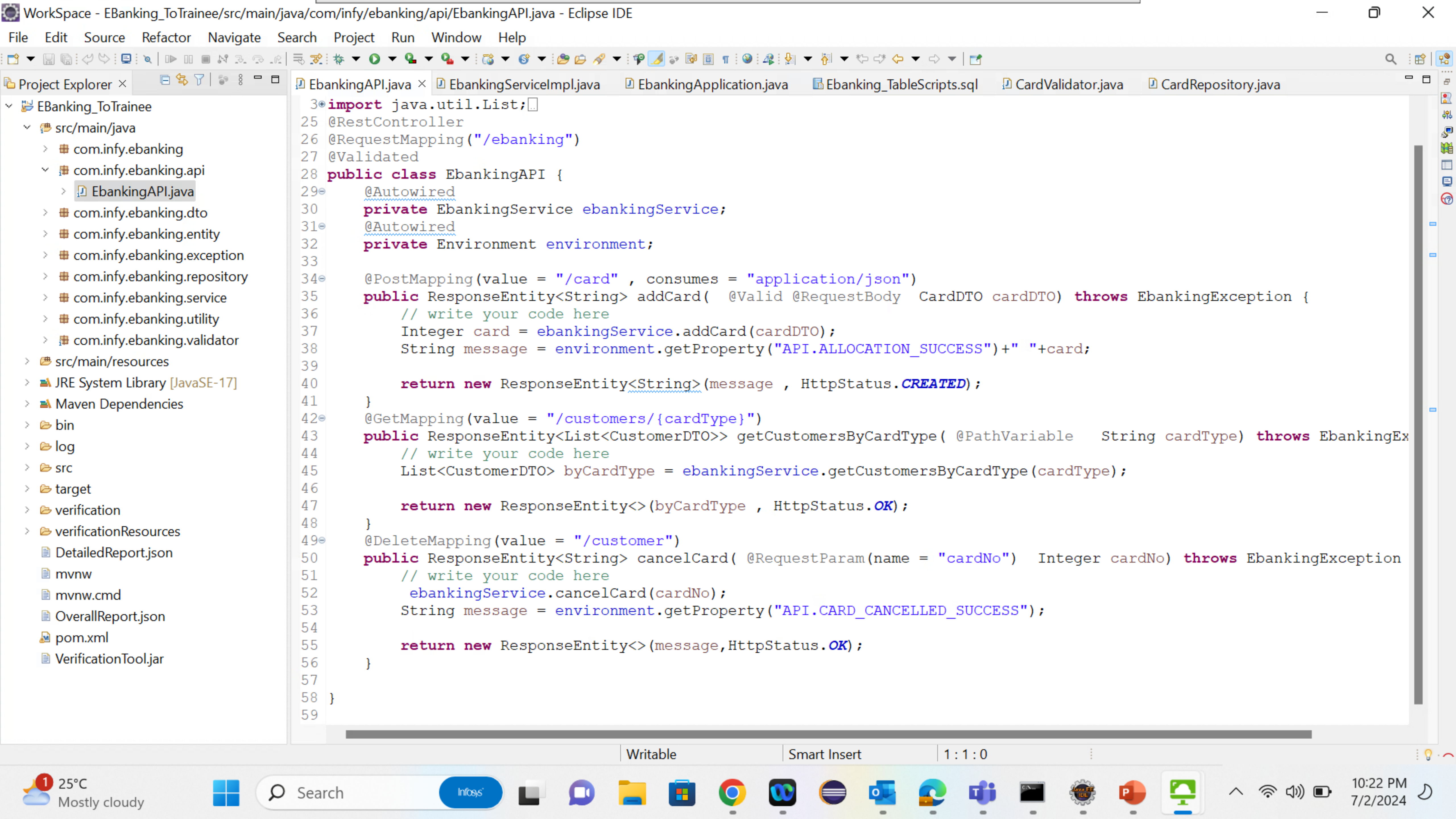
```java
package com.infy.ebanking.entity;

import java.time.LocalDate;

@Entity
@Table(name = "card")
public class Card {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer cardNo;
    private String cardType;
    private Integer minBalance;
    private LocalDate expiryDate;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "customer_id")
    private Customer customer;

    public Integer getCardNo() {
        return cardNo;
    }

    public void setCardNo(Integer cardNo) {
        this.cardNo = cardNo;
    }

    public String getCardType() {
        return cardType;
```

Validator Class



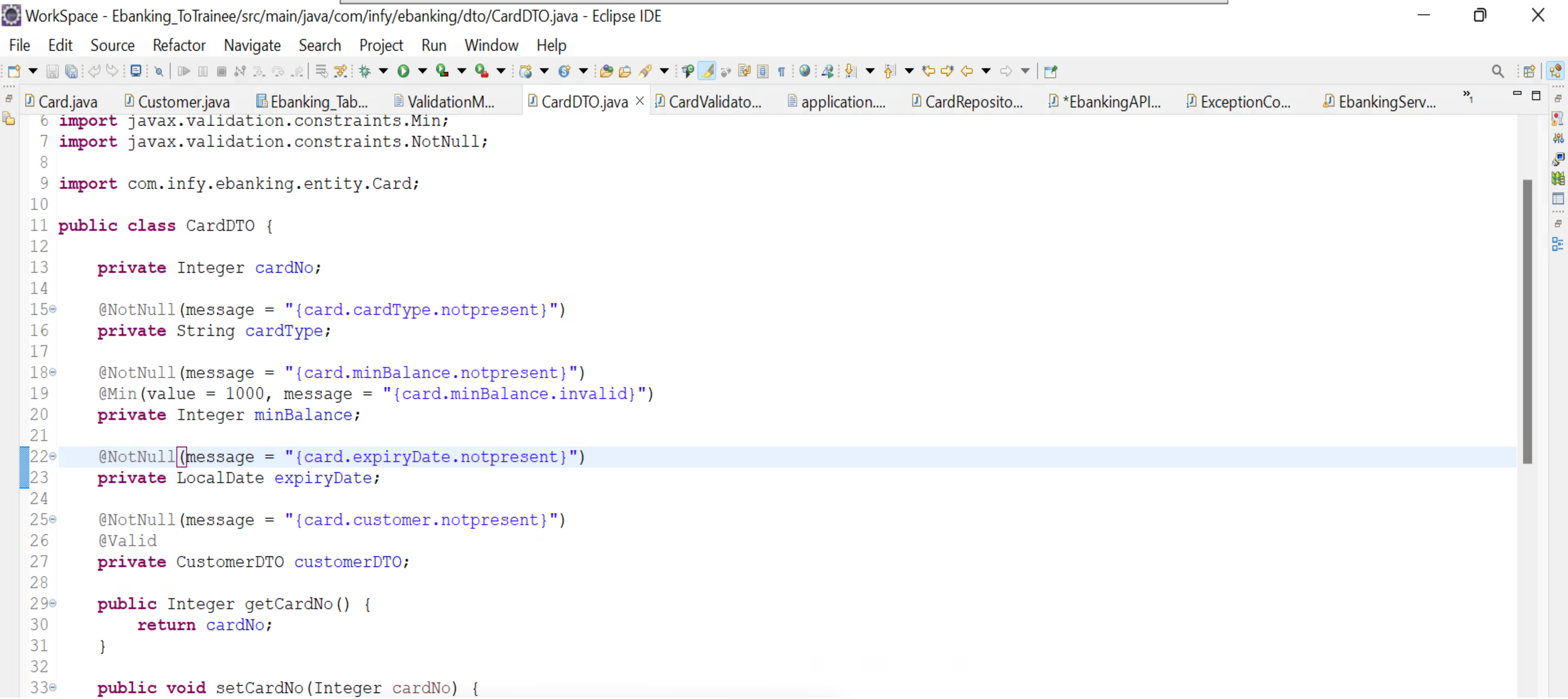
```java
package com.infy.ebanking.validator;

import com.infy.ebanking.dto.CardDTO;

public class CardValidator {

    public static void validate(CardDTO cardDTO) throws EbankingException {
        // write your code here
        if (Boolean.FALSE.equals((isValidCardType(cardDTO.getCardType())))) {
            throw new EbankingException("Validator.INVALID_CARDTYPE");
        }
    }

    public static Boolean isValidCardType(String cardType) {
        // write your code here
        String regex ="platinum|gold|silver";
        if(cardType==null) {
            return false;
        } else if(!cardType.matches(regex)){
            return false;
            }
        else {
            return true;
        }
        }
}
```

API Class



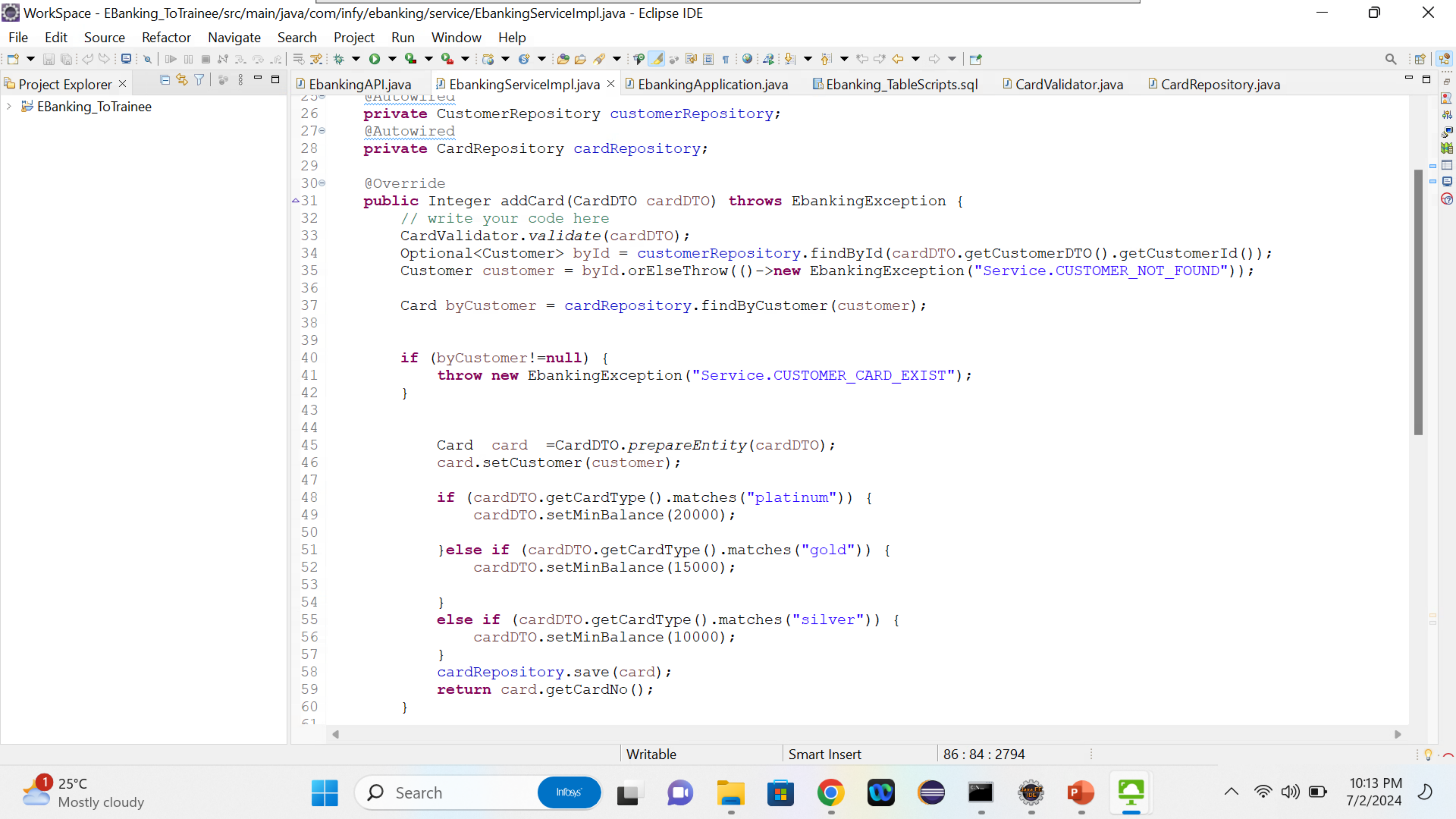
```java
import java.util.List;
@RestController
@RequestMapping("/ebanking")
@Validated
public class EbankingAPI {
    @Autowired
    private EbankingService ebankingService;
    @Autowired
    private Environment environment;

    @PostMapping(value = "/card" , consumes = "application/json")
    public ResponseEntity<String> addCard(  @Valid @RequestBody  CardDTO cardDTO) throws EbankingException {
        // write your code here
        Integer card = ebankingService.addCard(cardDTO);
        String message = environment.getProperty("API.ALLOCATION_SUCCESS")+" "+card;

        return new ResponseEntity<String>(message , HttpStatus.CREATED);
    }
    @GetMapping(value = "/customers/{cardType}")
    public ResponseEntity<List<CustomerDTO>> getCustomersByCardType( @PathVariable   String cardType) throws EbankingEx
        // write your code here
        List<CustomerDTO> byCardType = ebankingService.getCustomersByCardType(cardType);

        return new ResponseEntity<>(byCardType , HttpStatus.OK);
    }
    @DeleteMapping(value = "/customer")
    public ResponseEntity<String> cancelCard( @RequestParam(name = "cardNo")  Integer cardNo) throws EbankingException
        // write your code here
         ebankingService.cancelCard(cardNo);
        String message = environment.getProperty("API.CARD_CANCELLED_SUCCESS");

        return new ResponseEntity<>(message,HttpStatus.OK);
    }

}
```

Card DTO

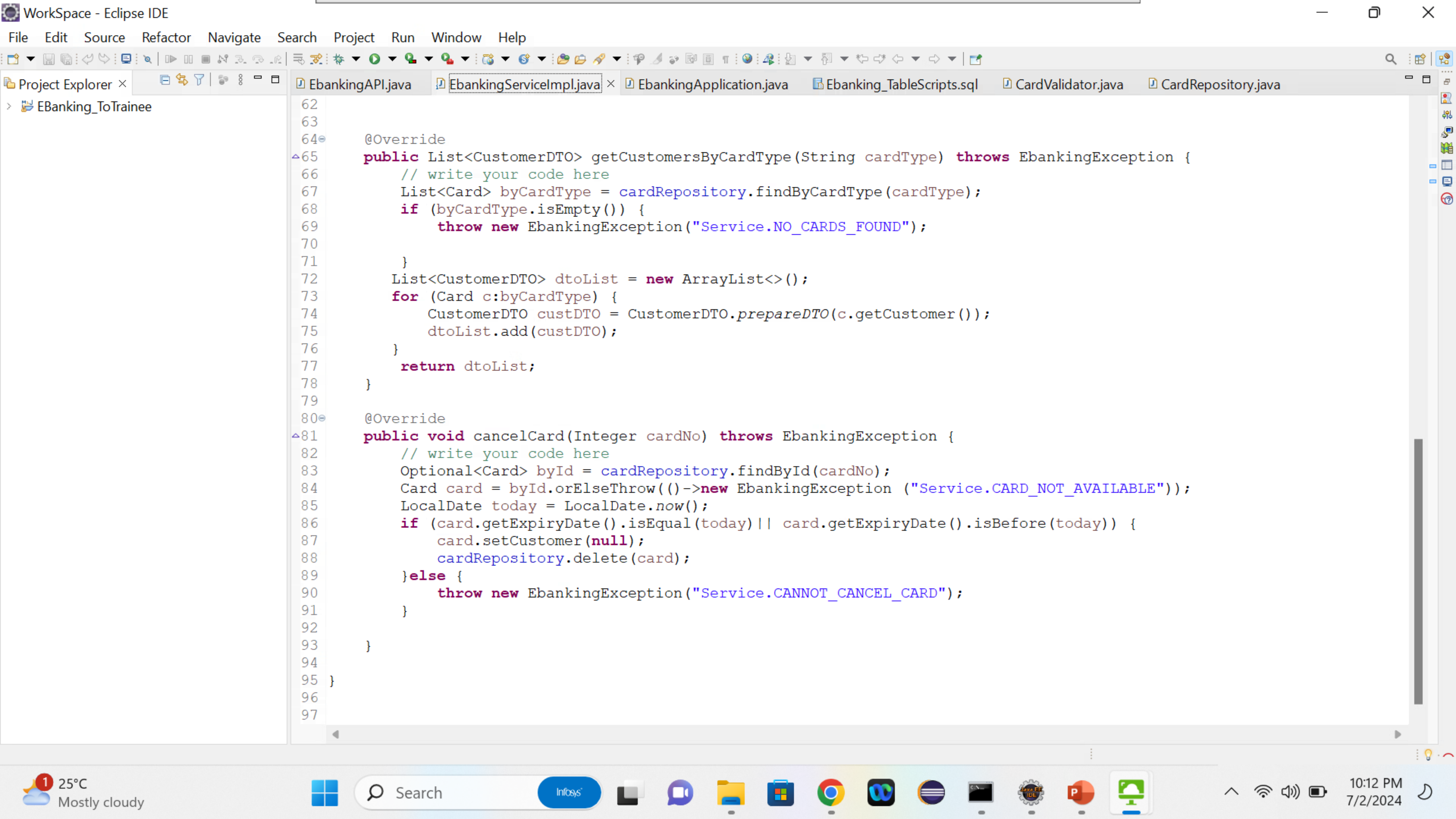


```java
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import com.infy.ebanking.entity.Card;

public class CardDTO {

    private Integer cardNo;

    @NotNull(message = "{card.cardType.notpresent}")
    private String cardType;

    @NotNull(message = "{card.minBalance.notpresent}")
    @Min(value = 1000, message = "{card.minBalance.invalid}")
    private Integer minBalance;

    @NotNull(message = "{card.expiryDate.notpresent}")
    private LocalDate expiryDate;

    @NotNull(message = "{card.customer.notpresent}")
    @Valid
    private CustomerDTO customerDTO;

    public Integer getCardNo() {
        return cardNo;
    }

    public void setCardNo(Integer cardNo) {
```

Service class of ebanking

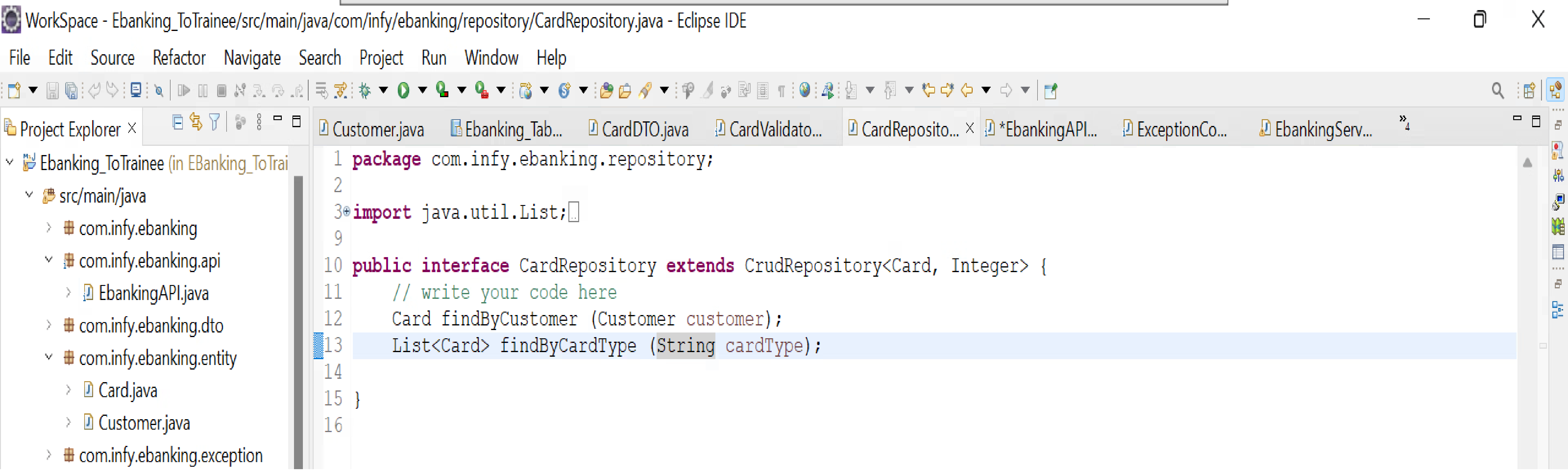


```java
    @Autowired
    private CustomerRepository customerRepository;
    @Autowired
    private CardRepository cardRepository;

    @Override
    public Integer addCard(CardDTO cardDTO) throws EbankingException {
        // write your code here
        CardValidator.validate(cardDTO);
        Optional<Customer> byId = customerRepository.findById(cardDTO.getCustomerDTO().getCustomerId());
        Customer customer = byId.orElseThrow(()->new EbankingException("Service.CUSTOMER_NOT_FOUND"));

        Card byCustomer = cardRepository.findByCustomer(customer);


        if (byCustomer!=null) {
            throw new EbankingException("Service.CUSTOMER_CARD_EXIST");
        }


            Card  card  =CardDTO.prepareEntity(cardDTO);
            card.setCustomer(customer);

            if (cardDTO.getCardType().matches("platinum")) {
                cardDTO.setMinBalance(20000);

            }else if (cardDTO.getCardType().matches("gold")) {
                cardDTO.setMinBalance(15000);

            }
            else if (cardDTO.getCardType().matches("silver")) {
                cardDTO.setMinBalance(10000);
            }
            cardRepository.save(card);
            return card.getCardNo();
        }
```



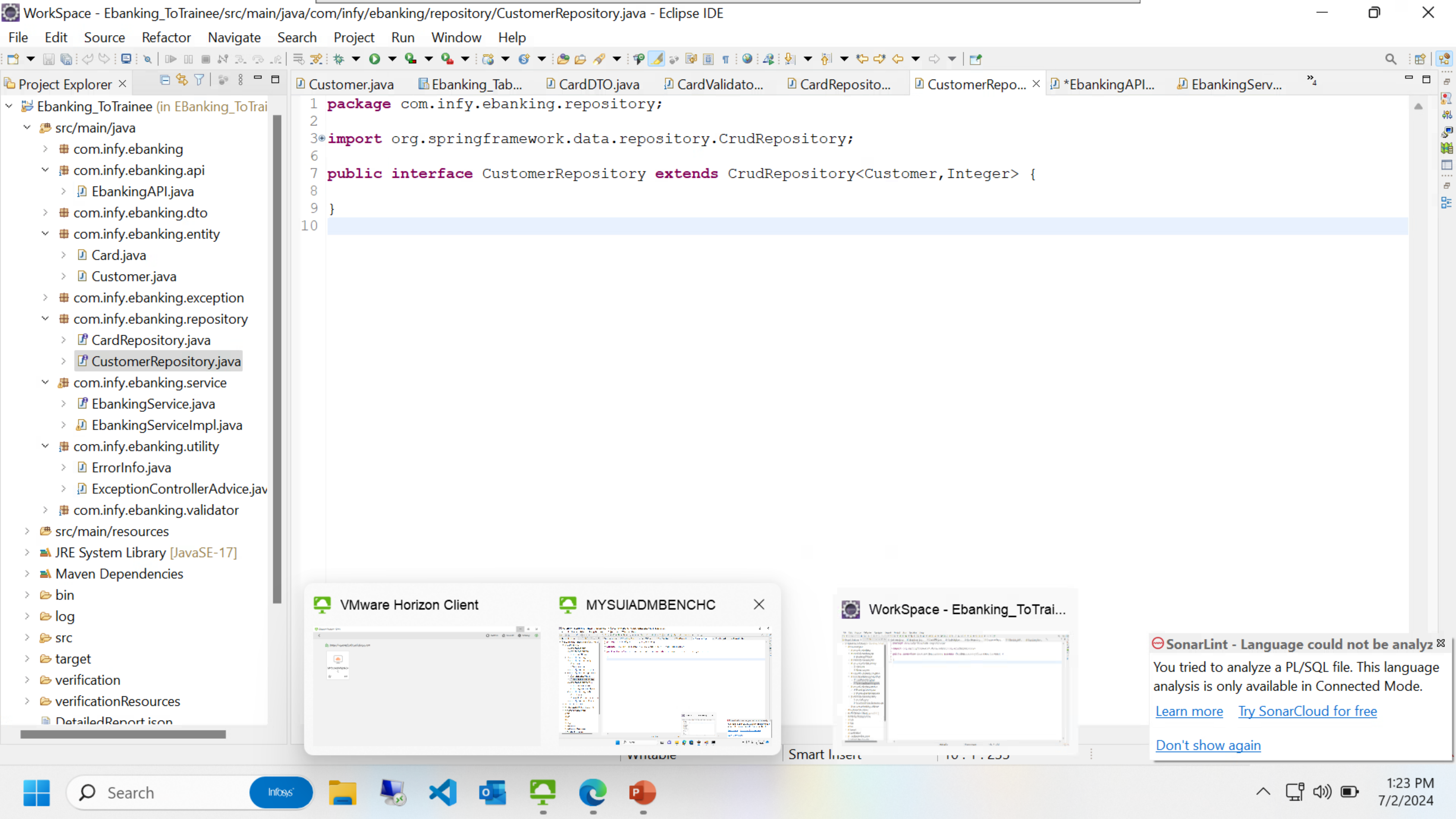
```
    @Override
    public List<CustomerDTO> getCustomersByCardType(String cardType) throws EbankingException {
        // write your code here
        List<Card> byCardType = cardRepository.findByCardType(cardType);
        if (byCardType.isEmpty()) {
            throw new EbankingException("Service.NO_CARDS_FOUND");

        }
       List<CustomerDTO> dtoList = new ArrayList<>();
       for (Card c:byCardType) {
           CustomerDTO custDTO = CustomerDTO.prepareDTO(c.getCustomer());
           dtoList.add(custDTO);
       }
        return dtoList;
    }

    @Override
    public void cancelCard(Integer cardNo) throws EbankingException {
        // write your code here
        Optional<Card> byId = cardRepository.findById(cardNo);
        Card card = byId.orElseThrow(()->new EbankingException ("Service.CARD_NOT_AVAILABLE"));
        LocalDate today = LocalDate.now();
        if (card.getExpiryDate().isEqual(today)|| card.getExpiryDate().isBefore(today)) {
            card.setCustomer(null);
            cardRepository.delete(card);
        }else {
            throw new EbankingException("Service.CANNOT_CANCEL_CARD");
        }

    }

}
```

Card Reposoitory



```java
package com.infy.ebanking.repository;

import java.util.List;

public interface CardRepository extends CrudRepository<Card, Integer> {
    // write your code here
    Card findByCustomer (Customer customer);
    List<Card> findByCardType (String cardType);

}
```

Customer Repository



```java
package com.infy.ebanking.repository;

import org.springframework.data.repository.CrudRepository;

public interface CustomerRepository extends CrudRepository<Customer,Integer> {

}
```

SonarLint - Language could not be analyz

You tried to analyze a PL/SQL file. This language analysis is only available in Connected Mode.

Learn more Try SonarCloud for free

Don't show again