Obiettivo: Implementare un sistema di notifiche flessibile e scalabile per una piattaforma di e-commerce, che possa gestire diversi tipi di notifiche e canali di comunicazione.

Classi e Interfacce:

1. `Notifica` (interfaccia)
   - Metodi: `invia()`, `getDestinatario()`, `getPriorita()`
2. `NotificaBase` (classe astratta che implementa `Notifica`)
   - Attributi: `destinatario`, `contenuto`, `timestamp`, `priorita`
   - Implementa metodi base comuni a tutte le notifiche
3. `NotificaEmail` (estende `NotificaBase`)
   - Attributi aggiuntivi: `oggetto`, `allegati`
4. `NotificaSMS` (estende `NotificaBase`)
   - Attributo aggiuntivo: `numeroCaratteri`
5. `NotificaPushApp` (estende `NotificaBase`)
   - Attributi aggiuntivi: `titolo`, `icona`
6. `NotificaWhatsApp` (estende `NotificaBase`)
   - Attributo aggiuntivo: `statoLettura`
7. `Utente`
   - Attributi: `id`, `nome`, `email`, `telefono`, `preferenzeNotifiche`
8. `EventoNotificabile` (interfaccia)
   - Metodi: `creaNotifica(Utente utente)`
9. `OrdineSpedito` (implementa `EventoNotificabile`)
10. `NuovaOfferta` (implementa `EventoNotificabile`)
11. `RecensioneRicevuta` (implementa `EventoNotificabile`)
12. `GestoreNotifiche`
    - Metodo per inviare notifiche
    - Metodo per gestire la coda di notifiche in base alla priorità
    - Metodo per tracciare lo stato di invio delle notifiche
13. `FactoryNotifiche`
    - Metodo per creare il tipo di notifica appropriato in base alle preferenze dell'utente e al tipo di evento

Implementazione base

```java
import java.time.LocalDateTime;
import java.util.*;
import java.util.concurrent.PriorityBlockingQueue;
import java.util.logging.Logger;

// Interfaccia Notifica
interface Notifica {
    void invia();
    String getDestinatario();
    int getPriorita();
}

// Classe astratta NotificaBase
abstract class NotificaBase implements Notifica {
    protected String destinatario;
    protected String contenuto;
    protected LocalDateTime timestamp;
    protected int priorita;

    public NotificaBase(String destinatario, String contenuto, int priorita) {
        this.destinatario = destinatario;
        this.contenuto = contenuto;
        this.timestamp = LocalDateTime.now();
        this.priorita = priorita;
    }

    @Override
    public String getDestinatario() {
        return destinatario;
    }

    @Override
    public int getPriorita() {
        return priorita;
    }
}

// Classi concrete per i diversi tipi di notifiche
class NotificaEmail extends NotificaBase {
    private String oggetto;
    private List<String> allegati;

    public NotificaEmail(String destinatario, String contenuto, int priorita,
String oggetto) {
        super(destinatario, contenuto, priorita);
        this.oggetto = oggetto;
```

```
        this.allegati = new ArrayList<>();
    }

    @Override
    public void invia() {
        System.out.println("Invio email a " + destinatario + ": " + oggetto);
    }
}

class NotificaSMS extends NotificaBase {
    private int numeroCaratteri;

    public NotificaSMS(String destinatario, String contenuto, int priorita) {
        super(destinatario, contenuto, priorita);
        this.numeroCaratteri = contenuto.length();
    }

    @Override
    public void invia() {
        System.out.println("Invio SMS a " + destinatario + " (" +
numeroCaratteri + " caratteri)");
    }
}

class NotificaPushApp extends NotificaBase {
    private String titolo;
    private String icona;

    public NotificaPushApp(String destinatario, String contenuto, int
priorita, String titolo, String icona) {
        super(destinatario, contenuto, priorita);
        this.titolo = titolo;
        this.icona = icona;
    }

    @Override
    public void invia() {
        System.out.println("Invio notifica push a " + destinatario + ": " +
titolo);
    }
}

class NotificaWhatsApp extends NotificaBase {
    private boolean statoLettura;

    public NotificaWhatsApp(String destinatario, String contenuto, int
```

```
priorita) {
        super(destinatario, contenuto, priorita);
        this.statoLettura = false;
    }

    @Override
    public void invia() {
        System.out.println("Invio messaggio WhatsApp a " + destinatario);
    }
}

// Classe Utente
class Utente {
    private String id;
    private String nome;
    private String email;
    private String telefono;
    private Set<String> preferenzeNotifiche;

    public Utente(String id, String nome, String email, String telefono) {
        this.id = id;
        this.nome = nome;
        this.email = email;
        this.telefono = telefono;
        this.preferenzeNotifiche = new HashSet<>();
    }

    // Getters e setters...
}

// Interfaccia EventoNotificabile
interface EventoNotificabile {
    Notifica creaNotifica(Utente utente);
}

// Implementazioni concrete degli eventi
class OrdineSpedito implements EventoNotificabile {
    @Override
    public Notifica creaNotifica(Utente utente) {
        return new NotificaEmail(utente.getEmail(), "Il tuo ordine è stato
spedito!", 2, "Ordine Spedito");
    }
}

class NuovaOfferta implements EventoNotificabile {
    @Override
```

```
    public Notifica creaNotifica(Utente utente) {
        return new NotificaPushApp(utente.getId(), "Nuova offerta
disponibile!", 3, "Offerta Speciale", "offer_icon.png");
    }
}

class RecensioneRicevuta implements EventoNotificabile {
    @Override
    public Notifica creaNotifica(Utente utente) {
        return new NotificaSMS(utente.getTelefono(), "Hai ricevuto una nuova
recensione!", 1);
    }
}

// Classe GestoreNotifiche
class GestoreNotifiche {
    private PriorityBlockingQueue<Notifica> codaNotifiche;
    private static final Logger logger =
Logger.getLogger(GestoreNotifiche.class.getName());

    public GestoreNotifiche() {
        this.codaNotifiche = new PriorityBlockingQueue<>(11,
Comparator.comparingInt(Notifica::getPriorita));
    }

    public void aggiungiNotifica(Notifica notifica) {
        codaNotifiche.offer(notifica);
        logger.info("Notifica aggiunta alla coda: " +
notifica.getClass().getSimpleName());
    }

    public void inviaNotifiche() {
        while (!codaNotifiche.isEmpty()) {
            Notifica notifica = codaNotifiche.poll();
            try {
                notifica.invia();
                logger.info("Notifica inviata: " +
notifica.getClass().getSimpleName() + " a " + notifica.getDestinatario());
            } catch (Exception e) {
                logger.warning("Errore nell'invio della notifica: " +
e.getMessage());
                // Qui si potrebbe implementare la logica di reinvio
            }
        }
    }
}
```

```
// Classe FactoryNotifiche
class FactoryNotifiche {
    public static Notifica creaNotifica(EventoNotificabile evento, Utente
utente) {
        return evento.creaNotifica(utente);
    }
}

// Classe principale per testare il sistema
public class SistemaNotifiche {
    public static void main(String[] args) {
        GestoreNotifiche gestore = new GestoreNotifiche();

        Utente utente1 = new Utente("1", "Mario Rossi", "mario@example.com",
"1234567890");
        Utente utente2 = new Utente("2", "Giulia Bianchi",
"giulia@example.com", "0987654321");

        EventoNotificabile ordineSpedito = new OrdineSpedito();
        EventoNotificabile nuovaOfferta = new NuovaOfferta();
        EventoNotificabile recensioneRicevuta = new RecensioneRicevuta();

        gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(ordineSpedito,
utente1));
        gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(nuovaOfferta,
utente2));

gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(recensioneRicevuta,
utente1));

        gestore.inviaNotifiche();
    }
}
```

Implementazione reale

```
import java.time.LocalDateTime;
import java.util.*;
import java.util.concurrent.*;
import java.util.logging.Logger;
import java.util.stream.Collectors;
import java.util.ResourceBundle;
```

```
interface Notifica {
    void invia();
    String getDestinatario();
    int getPriorita();
    String getTipo();
    LocalDateTime getTimestamp();
    String getContenuto();
}

abstract class NotificaBase implements Notifica {
    protected String destinatario;
    protected String contenuto;
    protected LocalDateTime timestamp;
    protected int priorita;
    protected StatoNotifica stato;

    public NotificaBase(String destinatario, String contenuto, int priorita) {
        this.destinatario = destinatario;
        this.contenuto = contenuto;
        this.timestamp = LocalDateTime.now();
        this.priorita = priorita;
        this.stato = StatoNotifica.IN_CODA;
    }

    @Override
    public String getDestinatario() { return destinatario; }
    @Override
    public int getPriorita() { return priorita; }
    @Override
    public LocalDateTime getTimestamp() { return timestamp; }
    @Override
    public String getContenuto() { return contenuto; }

    public StatoNotifica getStato() { return stato; }
    public void setStato(StatoNotifica stato) { this.stato = stato; }
}

class NotificaEmail extends NotificaBase {
    private String oggetto;
    private List<String> allegati;

    public NotificaEmail(String destinatario, String contenuto, int priorita,
String oggetto) {
        super(destinatario, contenuto, priorita);
        this.oggetto = oggetto;
        this.allegati = new ArrayList<>();
```

```
    }

    @Override
    public void invia() {
        System.out.println("Invio email a " + destinatario + ": " + oggetto);
        this.setStato(StatoNotifica.INVIATA);
    }

    @Override
    public String getTipo() { return "EMAIL"; }
}

// Implementazioni simili per NotificaSMS, NotificaPushApp, NotificaWhatsApp

enum StatoNotifica {
    IN_CODA, INVIATA, CONSEGNATA, FALLITA
}

class Utente {
    private String id;
    private String nome;
    private String email;
    private String telefono;
    private Set<String> preferenzeNotifiche;
    private String lingua;

    public Utente(String id, String nome, String email, String telefono,
String lingua) {
        this.id = id;
        this.nome = nome;
        this.email = email;
        this.telefono = telefono;
        this.preferenzeNotifiche = new HashSet<>();
        this.lingua = lingua;
    }

    // Getters e setters
    public String getId() { return id; }
    public String getEmail() { return email; }
    public String getTelefono() { return telefono; }
    public Set<String> getPreferenzeNotifiche() { return preferenzeNotifiche;
}
    public String getLingua() { return lingua; }
}

interface EventoNotificabile {
```

```
    Notifica creaNotifica(Utente utente, ResourceBundle bundle);
}

class OrdineSpedito implements EventoNotificabile {
    @Override
    public Notifica creaNotifica(Utente utente, ResourceBundle bundle) {
        String contenuto = bundle.getString("ordine.spedito");
        return new NotificaEmail(utente.getEmail(), contenuto, 2,
bundle.getString("ordine.spedito.oggetto"));
    }
}

// Implementazioni simili per NuovaOfferta, RecensioneRicevuta

class GestoreNotifiche {
    private PriorityBlockingQueue<Notifica> codaNotifiche;
    private static final Logger logger =
Logger.getLogger(GestoreNotifiche.class.getName());
    private Map<String, Integer> conteggiNotifiche;
    private static final int LIMITE_NOTIFICHE = 10;
    private ScheduledExecutorService scheduler;

    public GestoreNotifiche() {
        this.codaNotifiche = new PriorityBlockingQueue<>(11,
Comparator.comparingInt(Notifica::getPriorita));
        this.conteggiNotifiche = new ConcurrentHashMap<>();
        this.scheduler = Executors.newScheduledThreadPool(1);
        scheduler.scheduleAtFixedRate(this::resetConteggiNotifiche, 1, 1,
TimeUnit.HOURS);
    }

    public void aggiungiNotifica(Notifica notifica) {
        String destinatario = notifica.getDestinatario();
        if (conteggiNotifiche.getOrDefault(destinatario, 0) <
LIMITE_NOTIFICHE) {
            codaNotifiche.offer(notifica);
            conteggiNotifiche.merge(destinatario, 1, Integer::sum);
            logger.info("Notifica aggiunta alla coda: " + notifica.getTipo());
        } else {
            logger.warning("Limite notifiche raggiunto per " + destinatario);
        }
    }

    public void inviaNotifiche() {
        while (!codaNotifiche.isEmpty()) {
            Notifica notifica = codaNotifiche.poll();
```

```
            try {
                notifica.invia();
                logger.info("Notifica inviata: " + notifica.getTipo() + " a "
+ notifica.getDestinatario());
                AnalyticsTracker.trackNotifica(notifica, "INVIATA");
            } catch (Exception e) {
                logger.warning("Errore nell'invio della notifica: " +
e.getMessage());
                if (notifica instanceof NotificaBase) {
                    ((NotificaBase) notifica).setStato(StatoNotifica.FALLITA);
                    reinviaNotifica(notifica);
                }
            }
        }
    }

    private void reinviaNotifica(Notifica notifica) {
        scheduler.schedule(() -> {
            logger.info("Tentativo di reinvio per " + notifica.getTipo());
            aggiungiNotifica(notifica);
        }, 5, TimeUnit.MINUTES);
    }

    private void resetConteggiNotifiche() {
        conteggiNotifiche.clear();
        logger.info("Conteggi notifiche resettati");
    }

    public List<Notifica> aggregaNotifiche() {
        Map<String, List<Notifica>> notifichePerDestinatario =
codaNotifiche.stream()
                .collect(Collectors.groupingBy(Notifica::getDestinatario));

        List<Notifica> notificheAggregate = new ArrayList<>();
        for (List<Notifica> notificheUtente :
notifichePerDestinatario.values()) {
            if (notificheUtente.size() > 3) {
                String contenutoAggregato = notificheUtente.stream()
                        .map(Notifica::getContenuto)
                        .collect(Collectors.joining("\n"));
                NotificaEmail notificaAggregata = new NotificaEmail(
                        notificheUtente.get(0).getDestinatario(),
                        contenutoAggregato,
                        1,
                        "Notifiche aggregate"
                );
```

```
                notificheAggregate.add(notificaAggregata);
            } else {
                notificheAggregate.addAll(notificheUtente);
            }
        }
        return notificheAggregate;
    }
}

class FactoryNotifiche {
    public static Notifica creaNotifica(EventoNotificabile evento, Utente
utente) {
        ResourceBundle bundle = ResourceBundle.getBundle("notifiche", new
Locale(utente.getLingua()));
        return evento.creaNotifica(utente, bundle);
    }
}

class AnalyticsTracker {
    public static void trackNotifica(Notifica notifica, String azione) {
        // Implementazione per tracciare le metriche delle notifiche
        System.out.println("Analytics: " + azione + " - " + notifica.getTipo()
+ " per " + notifica.getDestinatario());
    }
}

public class SistemaNotifiche {
    public static void main(String[] args) {
        GestoreNotifiche gestore = new GestoreNotifiche();

        Utente utente1 = new Utente("1", "Mario Rossi", "mario@example.com",
"1234567890", "it");
        Utente utente2 = new Utente("2", "Giulia Bianchi",
"giulia@example.com", "0987654321", "en");

        EventoNotificabile ordineSpedito = new OrdineSpedito();
        EventoNotificabile nuovaOfferta = new NuovaOfferta();
        EventoNotificabile recensioneRicevuta = new RecensioneRicevuta();

        gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(ordineSpedito,
utente1));
        gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(nuovaOfferta,
utente2));

gestore.aggiungiNotifica(FactoryNotifiche.creaNotifica(recensioneRicevuta,
utente1));
```

```
        List<Notifica> notificheAggregate = gestore.aggregaNotifiche();
        for (Notifica notifica : notificheAggregate) {
            gestore.aggiungiNotifica(notifica);
        }

        gestore.inviaNotifiche();
    }
}
```