🌸 VALENTINE'S DAY CARD 🌸

Estimated time: 30 minutes
Learning Goals: Understand and practice inheritance, call parent constructor using super(), override methods with @Override, understand protected variables and see polymorphism in action.

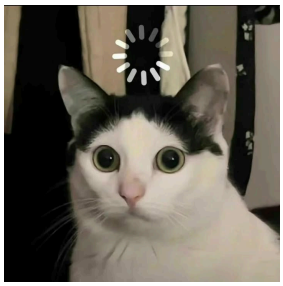
A Little About Valentine's Day

Valentine's Day (February 14th) is celebrated in many parts of the world as a day to appreciate the people we care about. While it's often associated with romantic love, it's also about friendship, kindness, and gratitude.

In Finland, we have Ystävänpäivä, which means "Friend's Day". So, instead of only focusing on couples, people celebrate all kinds of relationships and yes, that includes friends, family, and anyone that makes your life a little bit better and brighter. At its heart, Valentine's Day is about saying "I appreciate you", celebrating friendship, spreading kindness and making someone smile.

In today's lab, you'll build something positive and creative - a digital Valentine's card. You'll practice inheritance, overriding and polymorphism. You are allowed to express your creativity with your card. It can be romantic, funny, poetic, nerdy or completely out of this world. However, keep in mind that it should be kind and wholesome. No rude words allowed!

Now, let's combine programming with a sprinkle of joy to make people smile. 💖

Maybe off-topic, but I can't resist sharing these pics of how I lowkey imagine you're thinking about this lab.

Part 1: Understand the given code (5-10 minutes)

You're given the parent class which is ValentineCard.java.

Before you start creating your child classes, try to understand the given code.

- Has recipient and cardColor which are both protected
- Constructor sets up the window
- Methods: getMessage(), getDecoration(), displayCard()

Part 2: Create YOUR OWN cards! (20 minutes)

You're required to create **TWO** new card classes that extend ValentineCard.

Some ideas for your card type:

- RomanticCard.java - For your crush/partner
- FriendshipCard.java - For friends
- FunnyCard.java - With puns for your homies
- PoetryCard.java - With a short poem
- GamerCard.java - With gaming references like ("Be my Player 2!")
- Your own creative idea!

Template you can follow:

```
package main;

import java.awt.*;

public class YourCard extends ValentineCard {
    // Add your own private variables here (optional)
    private String yourVariable;


     //Constructor - MUST call super() first!
    public YourCard(String recipient /* add more parameters if you want
*/) {
        super(recipient); // THIS MUST BE FIRST LINE

        // Set your own card color (any color you want)
```

```
        this.cardColor = new Color(255, 204, 229); // or new Color(r, g,
b)
        getContentPane().setBackground(cardColor);

        // Initialize your variables
        this.yourVariable = "something";
    }

   // Write your own creative message here!
    @Override
    public String getMessage() {
        return "Your creative message here!<br>You can use HTML tags!";
    }

    // Put some decorations here with emojis
    @Override
    public String getDecoration() {
        return "🎮 💖 🎮"; // Use any emojis you want!
    }
}
```

Requirements:

- Class name ends with “Card”
- Extends ValentineCard
- Constructor calls super(recipient) as first line
- Changes cardColor to something different
- Overrides getMessage() with your own creative ones
- Overrides getDecoration() with your emojis
- Uses @Override annotation

Tips:

- You can use HTML syntax when creating your message.
-
 = new line
- My message for bigger text
- Link to various emojis you can use: https://emojipedia.org/
- Link to RGB calculator: https://www.w3schools.com/colors/colors_rgb.asp
- Link to Color Codes Chart: https://www.rapidtables.com/web/color/RGB_Color.html

Part 3: Display Your Card (5 minutes)

In your **App.java**, create your card objects and display your cards.

You can follow the template below:

```
package main;

import javax.swing.SwingUtilities;

public class App {

    public static void main(String[] args) {

        // Always start Swing applications like this
        SwingUtilities.invokeLater(() -> {

            // ================================
            // TODO 1: Create your card objects
            // ================================

            // Example:
            // ValentineCard card1 = new ValentineCard("Someone");

            // TODO: Create at least TWO more cards

            // ================================
            // TODO 2: Display your cards
            // ================================

            // For each card:
            // 1. Call displayCard()
            // 2. Call setVisible(true)
            // 3. Set a screen location


            // TODO: Display your other cards

            // ================================
```

```
            // TODO 3: Polymorphism Demo
            // ============================

            System.out.println("=== Polymorphism Demo ===");

            // TODO: Create an array of ValentineCard
            // Store ALL your card objects inside it


            // TODO: Loop through the array
            // Print each card's message using getMessage()

        });
    }
}
```

When you run App.java, you should get something like this: