```java
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.util.*;
import java.io.*;
import java.util.List;
import javax.imageio.ImageIO;

public class PokemonCardTracker {
    static class AttackInfo {
        String attackName;
        String damage;
        String effect;
        AttackInfo(String attackName, String damage, String effect) {
            this.attackName = attackName;
            this.damage = damage;
            this.effect = effect;
        }
        public String toString() { return attackName; }
    }

    static class Card {
        String name, type, attack, damage, rarity, effect;
        ImageIcon image;

        Card(String name, String type, String attack, String damage, String rarity, String effect, ImageIcon image) {
            this.name = name;
            this.type = type;
            this.attack = attack;
            this.damage = damage;
            this.rarity = rarity;
            this.effect = effect;
            this.image = image;
        }
    }

    // Direktorij Jar file-a
    public static String getJarDir() {
        try {
            String path = PokemonCardTracker.class.getProtectionDomain().getCodeSource().getLocation().toURI().getPath();
            File jarFile = new File(path);
            return jarFile.getParent();
        } catch (Exception e) {
            return System.getProperty("user.dir");
        }
    }

    // Custom JPanel za pozadinu app-a
    static class BackgroundPanel extends JPanel {
        private Image backgroundImage;
        public BackgroundPanel(String imagePath) {
            try {
                backgroundImage = ImageIO.read(new File(imagePath));
            } catch (IOException e) {
                System.out.println("Background image not found: " + e.getMessage());
            }
            setLayout(new BorderLayout());
        }
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (backgroundImage != null) {
                g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
            }
        }
    }

    // Podaci za atribute pokemona
    private static final Set<String> FIRST_GEN_POKEMON = new LinkedHashSet<>();
    private static final Set<String> POKEMON_TYPES = new LinkedHashSet<>();
    private static final Map<String, List<AttackInfo>> ATTACKS = new HashMap<>();
    private static final Map<String, String> POKEMON_TYPE = new HashMap<>();
    private static final String[] RARITIES = {
            "Common", "Uncommon", "Rare", "Rare Holo", "Reverse Holo", "Ultra Rare", "Secret Rare", "Promo", "Shiny Rare"
    };

    // Učitavanje podataka iz CSV-a
    static {
        String baseDir = getJarDir();
        File csvFile = new File(baseDir, "poke.csv");
        try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {
            String line = br.readLine(); // skip header
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",", 5);
                if (parts.length < 5) continue;
                String name = parts[0].trim();
                String type = parts[1].trim();
                String attack = parts[2].trim();
                String damage = parts[3].trim();
                String effect = parts[4].trim();
                FIRST_GEN_POKEMON.add(name);
                POKEMON_TYPES.add(type);
                POKEMON_TYPE.put(name, type);
                ATTACKS.computeIfAbsent(name, k -> new ArrayList<>())
                        .add(new AttackInfo(attack, damage, effect));
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error loading poke.csv: " + e.getMessage());
        }
    }

    private final List<Card> cards = new ArrayList<>();
    private final DefaultTableModel tableModel;
    private final JTable table;

    public PokemonCardTracker() {
        JFrame frame = new JFrame("Pokémon Card Tracker");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(1000, 550);

        // Jos jedan dio za custom pozadinu
        String baseDir = getJarDir();
        BackgroundPanel backgroundPanel = new BackgroundPanel(new File(baseDir, "pokeball.jpg").getAbsolutePath());

        // Postavljanje tablica
        String[] columns = {"Name", "Type", "Attack", "Damage", "Rarity", "Effect", "Image"};
        tableModel = new DefaultTableModel(columns, 0) {
            public boolean isCellEditable(int row, int column) { return false; }
            public Class<?> getColumnClass(int column) {
                return column == 6 ? ImageIcon.class : String.class;
            }
        };
        table = new JTable(tableModel);
        table.setRowHeight(60);
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setOpaque(false);
        scrollPane.getViewport().setOpaque(false);

        // Input polja
        JComboBox<String> nameBox = new JComboBox<>(FIRST_GEN_POKEMON.toArray(new String[0]));
```

```java
        JComboBox<String> typeBox = new JComboBox<>(POKEMON_TYPES.toArray(new String[0]));
        JComboBox<AttackInfo> attackBox = new JComboBox<>();
        JTextField damageField = new JTextField(4);
        damageField.setEditable(false);
        JComboBox<String> rarityBox = new JComboBox<>(RARITIES);
        JTextField effectField = new JTextField(18);
        JLabel imageLabel = new JLabel();
        JButton uploadBtn = new JButton("Upload Image");
        JButton addBtn = new JButton("Add Card");
        JButton removeBtn = new JButton("Remove Card");

        JPanel inputPanel = new JPanel();
        inputPanel.setOpaque(false);
        inputPanel.add(new JLabel("Name:")); inputPanel.add(nameBox);
        inputPanel.add(new JLabel("Type:")); inputPanel.add(typeBox);
        inputPanel.add(new JLabel("Attack:")); inputPanel.add(attackBox);
        inputPanel.add(new JLabel("Damage:")); inputPanel.add(damageField);
        inputPanel.add(new JLabel("Rarity:")); inputPanel.add(rarityBox);
        inputPanel.add(new JLabel("Effect:")); inputPanel.add(effectField);
        inputPanel.add(uploadBtn); inputPanel.add(imageLabel);
        inputPanel.add(addBtn);
        inputPanel.add(removeBtn);

        // Filter polja
        JComboBox<String> filterNameBox = new JComboBox<>();
        filterNameBox.addItem("");
        for (String n : FIRST_GEN_POKEMON) filterNameBox.addItem(n);
        JComboBox<String> filterTypeBox = new JComboBox<>();
        filterTypeBox.addItem("");
        for (String t : POKEMON_TYPES) filterTypeBox.addItem(t);
        JComboBox<String> filterAttackBox = new JComboBox<>();
        filterAttackBox.addItem("");
        JTextField filterDamage = new JTextField(4);
        JComboBox<String> filterRarityBox = new JComboBox<>();
        filterRarityBox.addItem("");
        for (String r : RARITIES) filterRarityBox.addItem(r);
        JTextField filterEffect = new JTextField(10);
        JButton filterBtn = new JButton("Filter");
        JButton resetBtn = new JButton("Reset");

        JPanel filterPanel = new JPanel();
        filterPanel.setOpaque(false);
        filterPanel.add(new JLabel("Name:")); filterPanel.add(filterNameBox);
        filterPanel.add(new JLabel("Type:")); filterPanel.add(filterTypeBox);
        filterPanel.add(new JLabel("Attack:")); filterPanel.add(filterAttackBox);
        filterPanel.add(new JLabel("Damage:")); filterPanel.add(filterDamage);
        filterPanel.add(new JLabel("Rarity:")); filterPanel.add(filterRarityBox);
        filterPanel.add(new JLabel("Effect:")); filterPanel.add(filterEffect);
        filterPanel.add(filterBtn); filterPanel.add(resetBtn);

        // Postavljanje napada kad se ime promjeni
        nameBox.addActionListener(e -> {
            String selected = (String) nameBox.getSelectedItem();
            attackBox.removeAllItems();
            if (selected != null && ATTACKS.containsKey(selected)) {
                for (AttackInfo info : ATTACKS.get(selected)) attackBox.addItem(info);
                typeBox.setSelectedItem(POKEMON_TYPE.get(selected));
            }
        });
        // Postavljanje efekta i damage-a kad se napad promjeni
        attackBox.addActionListener(e -> {
            AttackInfo info = (AttackInfo) attackBox.getSelectedItem();
            if (info != null) {
                damageField.setText(info.damage);
                effectField.setText(info.effect);
            } else {
                damageField.setText("");
                effectField.setText("");
            }
        });
        // Inicijalizacija attackboxa za prvog pokemona
        nameBox.setSelectedIndex(0);

        // Image upload
        final ImageIcon[] uploadedImage = {null};
        uploadBtn.addActionListener(e -> {
            JFileChooser fc = new JFileChooser();
            if (fc.showOpenDialog(frame) == JFileChooser.APPROVE_OPTION) {
                ImageIcon icon = new ImageIcon(fc.getSelectedFile().getAbsolutePath());
                Image img = icon.getImage().getScaledInstance(50, 60, Image.SCALE_SMOOTH);
                uploadedImage[0] = new ImageIcon(img);
                imageLabel.setIcon(uploadedImage[0]);
            }
        });

        // Dio za dodavanje karte
        addBtn.addActionListener(e -> {
            String name = (String) nameBox.getSelectedItem();
            String type = (String) typeBox.getSelectedItem();
            AttackInfo attackInfo = (AttackInfo) attackBox.getSelectedItem();
            String attack = attackInfo != null ? attackInfo.attackName : "";
            String damage = attackInfo != null ? attackInfo.damage : "";
            String rarity = (String) rarityBox.getSelectedItem();
            String effect = effectField.getText();
            ImageIcon img = uploadedImage[0];
            if (name == null || type == null || attack.isEmpty() || damage.isEmpty() || rarity == null || effect.isEmpty() || img == null) {
                JOptionPane.showMessageDialog(frame, "Fill all fields and upload an image.");
                return;
            }
            Card card = new Card(name, type, attack, damage, rarity, effect, img);
            cards.add(card);
            tableModel.addRow(new Object[]{name, type, attack, damage, rarity, effect, img});
            nameBox.setSelectedIndex(0); imageLabel.setIcon(null); uploadedImage[0]=null;
        });

        // Dio za micanje karte
        removeBtn.addActionListener(e -> {
            int selectedRow = table.getSelectedRow();
            if (selectedRow != -1) {
                cards.remove(selectedRow);
                tableModel.removeRow(selectedRow);
            } else {
                JOptionPane.showMessageDialog(frame, "Please select a card to remove.");
            }
        });

        // Filtering: ažuriranje napada kad se dropdown promjeni
        filterNameBox.addActionListener(e -> {
            String selected = (String) filterNameBox.getSelectedItem();
            filterAttackBox.removeAllItems();
            filterAttackBox.addItem("");
            if (selected != null && ATTACKS.containsKey(selected)) {
                for (AttackInfo info : ATTACKS.get(selected)) filterAttackBox.addItem(info.attackName);
            }
        });

        // Filtriranje karata
        filterBtn.addActionListener(e -> {
            String fName = (String) filterNameBox.getSelectedItem();
            String fType = (String) filterTypeBox.getSelectedItem();
            String fAttack = (String) filterAttackBox.getSelectedItem();
            String fDamage = filterDamage.getText().trim();
            String fRarity = (String) filterRarityBox.getSelectedItem();
            String fEffect = filterEffect.getText().trim().toLowerCase();
            tableModel.setRowCount(0);
            for (Card c : cards) {
                boolean matches = (fName == null || fName.isEmpty() || c.name.equals(fName)) &&
                        (fType == null || fType.isEmpty() || c.type.equals(fType)) &&
                        (fAttack == null || fAttack.isEmpty() || c.attack.equals(fAttack)) &&
                        (fDamage.isEmpty() || c.damage.equals(fDamage)) &&
                        (fRarity == null || fRarity.isEmpty() || c.rarity.equals(fRarity)) &&
                        (fEffect.isEmpty() || c.effect.toLowerCase().contains(fEffect));
                if (matches) {
                    tableModel.addRow(new Object[]{c.name, c.type, c.attack, c.damage, c.rarity, c.effect, c.image});
                }
            }
        });

        // Resetiranje filtera
        resetBtn.addActionListener(e -> {
            filterNameBox.setSelectedIndex(0); filterTypeBox.setSelectedIndex(0);
            filterAttackBox.removeAllItems(); filterAttackBox.addItem("");
            filterDamage.setText(""); filterRarityBox.setSelectedIndex(0); filterEffect.setText("");
            tableModel.setRowCount(0);
            for (Card c : cards)
                tableModel.addRow(new Object[]{c.name, c.type, c.attack, c.damage, c.rarity, c.effect, c.image});
        });

        // Dodavanje na panel sa backgroundom
        backgroundPanel.add(inputPanel, BorderLayout.NORTH);
        backgroundPanel.add(scrollPane, BorderLayout.CENTER);
        backgroundPanel.add(filterPanel, BorderLayout.SOUTH);

        frame.setContentPane(backgroundPanel);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(
PokemonCardTracker::new);
    }
}

```