

```
public class Participant
{
    // Les informations nécessaires

    private String firstname;
    private String name;
    private int birthYear;

    // ATTENTION: firstname peut être null !!!!!

    public Participant(String pFirstname, String pName, int pBirthYear)
    {
        firstname = pFirstname;
        name      = pName;
        birthYear = pBirthYear;
    }

    public String getFirstname()
    {
        return firstname;
    }

    public String getName()
    {
        return name;
    }

    public int getBirthYear()
    {
        return birthYear;
    }

    public String toString()
    {
        String n;
        if (firstname == null)
            n = "??";
        else
            n = firstname;
        return n + " " + name + " (" + birthYear + ")";
    }
}
```

```
public class Participants
{
    //
    // ATTENTION attribut firstname peut être null !!!!
    //

    // Liste des participants au cours
    private ArrayList<Participant> alParticipants = new ArrayList<>();

    public void add(Participant pParticipant)
    {
        alParticipants.add(pParticipant);
    }

    public void remove(int i)
    {
        alParticipants.remove(i);
    }

    public int size()
    {
        return alParticipants.size();
    }

    public Participant get(int pPosition)
    {
        return alParticipants.get(pPosition);
    }

    public Object[] toArray()
    {
        return alParticipants.toArray();
    }

    public Participant getEldest()
    {
        Participant current, eldest;
        if (size() == 0)
            return null;

        eldest = alParticipants.get(0);
        for (int i = 1; i < size(); i++)
        {
            current = alParticipants.get(i);
            if (current.getBirthYear() < eldest.getBirthYear())
                eldest = current;
        }

        return eldest;
    }

    public Participant getYoungest()
    {
        Participant current, youngest;
        if (size() == 0)
            return null;

        youngest = alParticipants.get(0);
        for (int i = 1; i < size(); i++)
        {
            current = alParticipants.get(i);
            if (current.getBirthYear() > youngest.getBirthYear())
                youngest = current;
        }

        return youngest;
    }

    // ... prochaine page ...
}
```

```
public int searchByBirthYear(int pYear)
{
    boolean found = false;

    int i = 0;
    while (!found && (i < alParticipants.size()))
    {
        if (pYear == alParticipants.get(i).getBirthYear())
            found = true;
        else
            i++;
    }

    if (found)
        return i;
    else
        return -1;
}

public int searchByName(String pName)
{
    boolean found = false;

    int i = 0;
    while (!found && (i < alParticipants.size()))
    {
        if (pName.equals(alParticipants.get(i).getName()))
            found = true;
        else
            i++;
    }

    if (found)
        return i;
    else
        return -1;
}

public int countTextInFirstName(String text)
{
    int count = 0;

    for (int i = 0; i < alParticipants.size(); i++)
    {
        String v = alParticipants.get(i).getFirstname();
        if ((v != null) && (v.contains(text)))
        {
            count++;
        }
    }

    return count;
}

public void removeOlder(int year)
{
    for (int i = alParticipants.size() - 1; i >= 0; i--)
    {
        if (alParticipants.get(i).getBirthYear() < year)
        {
            alParticipants.remove(i);
        }
    }
}

public Participant searchLastByName(String pName)
{
    boolean found = false;

    int i = alParticipants.size() - 1;
    while (!found && (i >= 0))
    {
        if (pName.equals(alParticipants.get(i).getName()))
            found = true;
        else
            i--;
    }

    if (found)
        return alParticipants.get(i);
    else
        return null;
}

public ArrayList<Participant> searchAllYounger(int pYear)
{
    ArrayList<Participant> res = new ArrayList<>();
    for (int i = 0; i < alParticipants.size(); i++)
    {
        if (alParticipants.get(i).getBirthYear() > pYear)
            res.add(alParticipants.get(i));
    }
    return res;
}

// ... prochaine page ...
```

```
public void sortByFirstNameAsc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        String min = alParticipants.get(i).getFirstname();
        if (min == null)
            min = "";
        int posMin = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            String v = alParticipants.get(j).getFirstname();
            if (v == null)
                v = "";

            if (v.compareTo(min) < 0)
            {
                min = v;
                posMin = j;
            }
        }

        if (i != posMin)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMin));
            alParticipants.set(posMin, temp);
        }
    }
}

public void sortByFirstNameDesc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        String max = alParticipants.get(i).getFirstname();
        if (max == null)
            max = "";
        int posMax = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            String v = alParticipants.get(j).getFirstname();
            if (v == null)
                v = "";
            if (v.compareTo(max) > 0)
            {
                max = v;
                posMax = j;
            }
        }

        if (i != posMax)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMax));
            alParticipants.set(posMax, temp);
        }
    }
}

public void sortByNameAsc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        String min = alParticipants.get(i).getName();
        int posMin = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            if (alParticipants.get(j).getName().compareTo(min) < 0)
            {
                min = alParticipants.get(j).getName();
                posMin = j;
            }
        }

        if (i != posMin)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMin));
            alParticipants.set(posMin, temp);
        }
    }
}
```

*// ... prochaine page ...*

```
public void sortByNameDesc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        String max = alParticipants.get(i).getName();
        int posMax = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            if (alParticipants.get(j).getName().compareTo(max) > 0)
            {
                max = alParticipants.get(j).getName();
                posMax = j;
            }
        }

        if (i != posMax)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMax));
            alParticipants.set(posMax, temp);
        }
    }
}

public void sortByYearAsc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        int min = alParticipants.get(i).getBirthYear();
        int posMin = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            if (alParticipants.get(j).getBirthYear() < min)
            {
                min = alParticipants.get(j).getBirthYear();
                posMin = j;
            }
        }

        if (i != posMin)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMin));
            alParticipants.set(posMin, temp);
        }
    }
}

public void sortByYearDesc()
{
    for (int i=0; i<alParticipants.size()-1; i++)
    {
        int max = alParticipants.get(i).getBirthYear();
        int posMax = i;
        for (int j=i+1; j<alParticipants.size(); j++)
        {
            if (alParticipants.get(j).getBirthYear() > max)
            {
                max = alParticipants.get(j).getBirthYear();
                posMax = j;
            }
        }

        if (i != posMax)
        {
            Participant temp = alParticipants.get(i);
            alParticipants.set(i, alParticipants.get(posMax));
            alParticipants.set(posMax, temp);
        }
    }
}
```

```
// ... prochaine page - pour les très avancés !!!!...
```

```
public void sortByNameAndFirstNameAsc()
{
    //
    // PAS À SAVOIR - Juste pour ceux que cela intéresse !!!!
    //

    // Commencer par trier par nom de famille
    sortByNameAsc();

    // pour les noms de famille identiques, trier par le prénom...
    String name;
    String next;
    int first = 0;
    while (first < alParticipants.size()-1)
    {
        name = alParticipants.get(first).getName();
        next = alParticipants.get(first+1).getName();
        if (name.equals(next))
        {
            // recherche les limites des noms identiques
            int last = first+1;
            boolean end = false;
            while (!end && (last < alParticipants.size()))
            {
                String v = alParticipants.get(last).getName();
                if (!v.equals(name))
                {
                    // les noms identiques sont terminés
                    end = true;
                }
                else
                    last++;
            }

            // trier de first à last (last non inclus)
            for (int i=first; i<last-1; i++)
            {
                String min = alParticipants.get(i).getFirstname();
                if (min == null)
                    min = "";
                int posMin = i;
                for (int j=i+1; j<last; j++)
                {
                    String v = alParticipants.get(j).getFirstname();
                    if (v == null)
                        v = "";
                    if (v.compareTo(min) < 0)
                    {
                        min = v;
                        posMin = j;
                    }
                }

                if (i != posMin)
                {
                    Participant temp = alParticipants.get(i);
                    alParticipants.set(i, alParticipants.get(posMin));
                    alParticipants.set(posMin, temp);
                }
            }
            // on passe à la suite
            first = last;
        }
        else
            first++;
    }
}
```

```

public class MainFrame extends javax.swing.JFrame
{
    private Participants participants = new Participants();

    public MainFrame()
    {
        initComponents();

        // TEST PURPOSES ONLY...
        participants.add(new Participant("Josy", "Muller", 1980));
        participants.add(new Participant("Ren", "Smith", 1966));
        participants.add(new Participant("Poli", "Smith", 1985));
        participants.add(new Participant(null, "Smith", 1999));
        participants.add(new Participant("", "Smith", 1965));
        participants.add(new Participant("Mary", "Nemo", 1978));
        participants.add(new Participant(null, "Nemo", 1898));
        participants.add(new Participant("John", "Zappa", 1943));

        updateView();
    }

    private void updateView()
    {
        participantsList.setListData(participants.toArray());

        // afficher un "-" si "eldest" ou "youngest" n'existe pas
        Participant eldest = participants.getEldest();
        if (eldest == null)
            eldestLabel.setText("-");
        else
            eldestLabel.setText(eldest.toString());

        Participant youngest = participants.getYoungest();
        if (youngest == null)
            youngestLabel.setText("-");
        else
            youngestLabel.setText(youngest.toString());
    }
// Skipped: ... initComponents { ... }
    private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_addButtonActionPerformed
        String firstname = firstnameTextField.getText();
        String name = nameTextField.getText();
        int age = Integer.valueOf(ageTextField.getText());

        Participant p = new Participant(firstname, name, age);
        participants.add(p);

        // Efface tous les champs
        nameTextField.setText("");
        firstnameTextField.setText("");
        ageTextField.setText("");

        // Améliorations: remet le curseur dans le champ 'prénom'
        firstnameTextField.requestFocus();

        updateView();
    } //GEN-LAST:event_addButtonActionPerformed

    private void removeButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_removeButtonActionPerformed
        // Vérifier s'il y a bien qqchse de sélectionné
        int index = participantsList.getSelectedIndex();
        if (index >= 0)
            participants.remove(index);
        updateView();
    } //GEN-LAST:event_removeButtonActionPerformed

    private void searchBirthyearButtonActionPerformed(java.awt.event.ActionEvent evt) //GEN-FIRST:event_searchBirthyearButtonActionPerformed
    { //GEN-HEADEREND:event_searchBirthyearButtonActionPerformed
        // désélectionne la liste
        participantsList.clearSelection();

        String ageVal = ageTextField.getText();
        if (ageVal.equals(""))
            return;

        int year = Integer.valueOf(ageVal);
        int pos = participants.searchByBirthYear(year);
        if (pos != -1)
            participantsList.setSelectedIndex(pos);
    } //GEN-LAST:event_searchBirthyearButtonActionPerformed

    private void searchNameButtonActionPerformed(java.awt.event.ActionEvent evt) //GEN-FIRST:event_searchNameButtonActionPerformed
    { //GEN-HEADEREND:event_searchNameButtonActionPerformed
        // désélectionne la liste
        participantsList.clearSelection();

        String name = nameTextField.getText();
        if (name.equals(""))
            return;

        int pos = participants.searchByName(name);
        if (pos != -1)
            participantsList.setSelectedIndex(pos);
    } //GEN-LAST:event_searchNameButtonActionPerformed
}

```

```

private void countFirstnameButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_countFirstnameButtonActionPerformed
{
    //GEN-HEADEREND:event_countFirstnameButtonActionPerformed
    // enlève le compteur s'il existait avant
    countLabel.setText("-");

    String firstName = firstNameTextField.getText();
    if (firstName.equals(""))
        return;
    int count = participants.countTextInFirstName(firstName);
    countLabel.setText("Count: " + count);
}
//GEN-LAST:event_countFirstnameButtonActionPerformed

private void removeOlderButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_removeOlderButtonActionPerformed
{
    //GEN-HEADEREND:event_removeOlderButtonActionPerformed
    String ageVal = ageTextField.getText();
    if (ageVal.equals(""))
        return;

    int year = Integer.valueOf(ageVal);
    participants.removeOlder(year);
    updateView();
}
//GEN-LAST:event_removeOlderButtonActionPerformed

private void findLastByNameButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_findLastByNameButtonActionPerformed
{
    //GEN-HEADEREND:event_findLastByNameButtonActionPerformed
    // enlève le résultat s'il existait avant
    lastLabel.setText("-");

    String name = nameTextField.getText();
    if (name.equals(""))
        return;

    Participant p = participants.searchLastByName(name);
    if (p != null)
        lastLabel.setText(p.toString());
}
//GEN-LAST:event_findLastByNameButtonActionPerformed

private void findAllButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_findAllButtonActionPerformed
{
    //GEN-HEADEREND:event_findAllButtonActionPerformed
    // enlève le résultat s'il existait avant
    nbrAllLabel.setText("-");

    String ageVal = ageTextField.getText();
    if (ageVal.equals(""))
        return;

    int year = Integer.valueOf(ageVal);
    ArrayList<Participant> res = participants.searchAllYounger(year);
    nbrAllLabel.setText("Number of participants found: " + res.size());
}
//GEN-LAST:event_findAllButtonActionPerformed

private void sortByFNameAscButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByFNameAscButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByFNameAscButtonActionPerformed
    participants.sortByFirstNameAsc();
    updateView();
}
//GEN-LAST:event_sortByFNameAscButtonActionPerformed

private void sortByFNameDescButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByFNameDescButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByFNameDescButtonActionPerformed
    participants.sortByFirstNameDesc();
    updateView();
}
//GEN-LAST:event_sortByFNameDescButtonActionPerformed

private void sortByNameAscButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByNameAscButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByNameAscButtonActionPerformed
    participants.sortByNameAsc();
    updateView();
}
//GEN-LAST:event_sortByNameAscButtonActionPerformed

private void sortByNameDescButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByNameDescButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByNameDescButtonActionPerformed
    participants.sortByNameDesc();
    updateView();
}
//GEN-LAST:event_sortByNameDescButtonActionPerformed

private void sortByYearAscButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByYearAscButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByYearAscButtonActionPerformed
    participants.sortByYearAsc();
    updateView();
}
//GEN-LAST:event_sortByYearAscButtonActionPerformed

private void sortByYearDescButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByYearDescButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByYearDescButtonActionPerformed
    participants.sortByYearDesc();
    updateView();
}
//GEN-LAST:event_sortByYearDescButtonActionPerformed

private void sortByNameAndFNameAscButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_sortByNameAndFNameAscButtonActionPerformed
{
    //GEN-HEADEREND:event_sortByNameAndFNameAscButtonActionPerformed
    participants.sortByNameAndFirstNameAsc();
    updateView();
}
//GEN-LAST:event_sortByNameAndFNameAscButtonActionPerformed
// Skipped: ... Look @ Feel
// Skipped: ... graphic attributes
}

```