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

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

     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()
     {
          return firstname + " " + name + " (*" + birthYear + ")";
     }
}
```

```java
import java.util.ArrayList;
public class Participants
{
     // PARTIE 2: algorithmes correctement programmés

     // 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 Participant getEldest()
     {
          int min;
          Participant current, eldest;

          if (alParticipants.isEmpty())
               return null;

          eldest = alParticipants.get(0);
          min = eldest.getBirthYear();

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

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

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

     public Participant getYoungest2()
     {
          // Version la plus élégante
          if (alParticipants.isEmpty())
               return null;

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

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

```java
public class TestParticipants
{
     /**
      * Programme principal.
      */
     public static void main(String[] args)
     {
          Participant josy = new Participant("Josy", "Muller", 1980);
          Participant poli = new Participant("Poli", "Klein", 1985);
          Participant mary = new Participant("Mary", "Nemo", 1978);
          Participants participants = new Participants();
          participants.add(josy);
          participants.add(poli);
          participants.add(mary);

          for (int i = 0; i < participants.size(); i++)
               System.out.println(participants.get(i));

          System.out.println("-------------");

          participants.remove(0);

          for (int i = 0; i < participants.size(); i++)
               System.out.println(participants.get(i));

          System.out.println("Eldest: " + participants.getEldest());
          System.out.println("Youngest: " + participants.getYoungest());
     }
}
```

```java
public class MainFrame extends javax.swing.JFrame
{
     // PARTIE 2: algorithmes correctement programmés

     private Participants participants = new Participants();

     public MainFrame()
     {
          initComponents();
          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 qqchose de sélectionné
          int index = participantsList.getSelectedIndex();
          if (index >= 0)
               participants.remove(index);
          updateView();
    }//GEN-LAST:event_removeButtonActionPerformed
// Skipped: ... Look & Feel
// Skipped: ... graphic attributes
}
```