



POLITECNICO  
DI MILANO

# INFORMATICA

Le classi nel modello ad  
ambienti

```
class Tree
{ public:
    Tree();
    void inserisciSeNonEsiste(Studente);
    void inOrderTraversal();
    void stampaAlberoRovesciato();
private:
    TreeNode *rootPtr;
    void inserisciConRicorsione(TreeNode *&, Studente);
    void recursiveInOrder(TreeNode *);
    void stampaRovesciatoRicorsiva(TreeNode *, int);
};

Tree :: Tree()
{ rootPtr = 0;
}

void Tree :: inserisciSeNonEsiste(Studente nuovoStudente)
{ inserisciConRicorsione(rootPtr, nuovoStudente);
}


void Tree :: inserisciConRicorsione(TreeNode *& ptr,
                                   Studente nuovoStudente)
{ if (ptr == 0)
    { ptr = new TreeNode(NuovoStudente);
    }
  else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->leftPtr, nuovoStudente);
  else if (nuovoStudente.matricola > ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->rightPtr, nuovoStudente);
  else
    cout << "studente di matricola " << nuovoStudente.matricola
          << " già presente" << endl;
}
```

```
class Tree
{ public:
    Tree();
    void inserisciSeNonEsiste(Studente);
    void inOrderTraversal();
    void stampaAlberoRovesciato();
private:
    TreeNode *rootPtr;
    void inserisciConRicorsione(TreeNode *&, Studente);
    void recursiveInOrder(TreeNode *);
    void stampaRovesciatoRicorsiva(TreeNode *, int);
};

Tree :: Tree()
{ rootPtr = 0;
}

void Tree :: inserisciSeNonEsiste(Studente nuovoStudente)
{ inserisciConRicorsione(rootPtr, nuovoStudente);
}

void Tree :: inserisciConRicorsione(TreeNode *& ptr,
                                     Studente nuovoStudente)
{ if (ptr == 0)
    { ptr = new TreeNode(NuovoStudente);
    }
  else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->leftPtr, nuovoStudente);
  else if (nuovoStudente.matricola > ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->rightPtr, nuovoStudente);
  else
    cout << "studente di matricola " << nuovoStudente.matricola
          << " già presente" << endl;
}
```



```

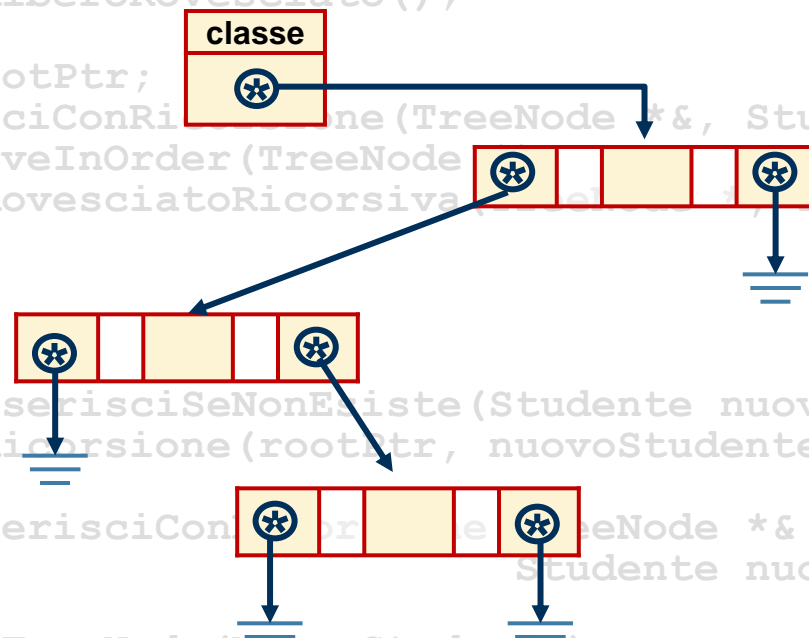
class Tree
{ public:
    Tree();
    void inserisciSeNonEsiste(Studiante);
    void inOrderTraversal();
    void stampaAlberoRovesciato();
private:
    TreeNode *rootPtr;
    void inserisciConRicorsione(TreeNode *&, Studiante);
    void recursiveInOrder(TreeNode *, int);
    void stampaRovesciatoRicorsiva(TreeNode *, int);
};

Tree :: Tree()
{ rootPtr = 0;
}

void Tree :: inserisciSeNonEsiste(Studiante nuovoStudiante)
{ inserisciConRicorsione(rootPtr, nuovoStudiante);
}

void Tree :: inserisciConRicorsione(TreeNode *& ptr,
                                     Studiante nuovoStudiante)
{ if (ptr == 0)
    { ptr = new TreeNode(nuovoStudiante);
    }
  else if (nuovoStudiante.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->leftPtr, nuovoStudiante);
  else if (nuovoStudiante.matricola > ptr->datiStud.matricola)
    inserisciConRicorsione(ptr->rightPtr, nuovoStudiante);
  else
    cout << "studente di matricola " << nuovoStudiante.matricola
          << " già presente" << endl;
}

```



```
int main()
{ const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Nodo *classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento,"
    << " e per eliminazione, f per fine" << " ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
```

```
int main()
{ const char INSERISCI = 'i';
  const char ELIMINA = 'e';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Nodo *classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento,"
    << " e per eliminazione, f per fine" << " ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Nodo *classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
```



```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Nodo *classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
```



```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  classe = inizializzaTabella();
  stampaTabella(classe);
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
    }
  }
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
    }

    << " e per eliminazione, f per fine" << " ): ";
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
    }

    << " e per eliminazione, f per fine" << " ): ";
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
    }

    << " e per eliminazione, f per fine" << " ): ";
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      inserisciSeNonEsiste(nuovoStudente, classe);
      stampaTabella(classe);
    }
  }
```

```
<< " e per eliminazione, f per fine" << " ): ";
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      classe.inserisciSeNonEsiste(nuovoStudente);
      stampaTabella(classe);
    }
  }
```

```
<< " e per eliminazione, f per fine" << " ): ";
```

```
int main()
{ const char INSERISCI = 'i';
  const char FINE = 'f';
  char operazione;
  Studente nuovoStudente;
  int matricola;
  Tree classe;
  cout << setw(50) << "acquisizione operazione" << endl << endl;
  cout << "operazione? ( i per inserimento, f per fine ): ";
  cin >> operazione;
  while (operazione != FINE)
  { if (operazione == INSERISCI)
    { cout << " dati del nuovo studente" << endl;
      cout << " matricola: "; cin >> nuovoStudente.matricola;
      cout << " cognome: "; cin >> nuovoStudente.cognome;
      cout << " nome: "; cin >> nuovoStudente.nome;
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;
      cout << " giorno di nascita (1-31): ";
      cin >> nuovoStudente.dataNascita.giorno;
      cout << " mese di nascita (1-12): ";
      cin >> nuovoStudente.dataNascita.mese;
      cout << " anno di nascita (0-99): ";
      cin >> nuovoStudente.dataNascita.anno;
      classe.inserisciSeNonEsiste(nuovoStudente);
    }
    cout << "operazione? ( i per inserimento,"
           << " e per eliminazione, f per fine" << " ): ";
    cin >> operazione
  }
```



```
classe TreeNode
```

```
{ ...  
};
```

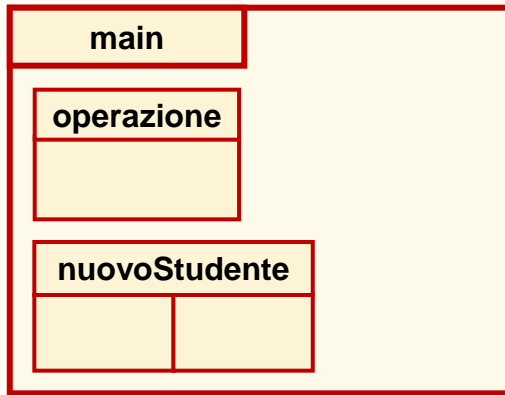
```
classe Tree
```

```
{ ...  
};
```

```
int main()
```

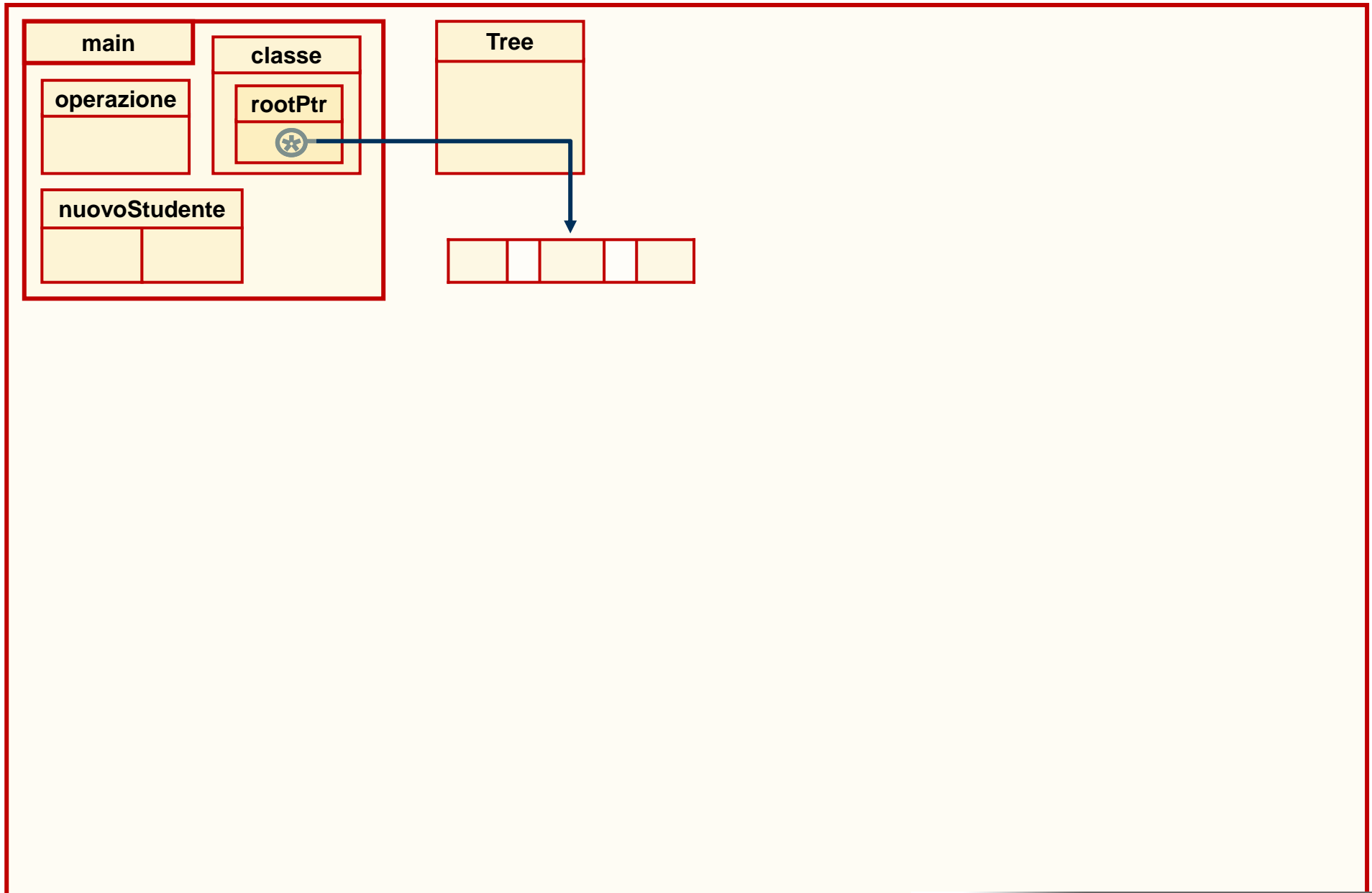
```
{ const char INSERISCI = 'i';  
  const char FINE = 'f';  
  char operazione;  
  Studente nuovoStudente;  
  int matricola;  
  Tree classe;  
  cout << setw(50) << "acquisizione operazione" << endl << endl;  
  cout << "operazione? ( i per inserimento, f per fine ): ";  
  cin >> operazione;  
  while (operazione != FINE)  
  { if (operazione == INSERISCI)  
    { cout << " dati del nuovo studente" << endl;  
      cout << " matricola: "; cin >> nuovoStudente.matricola;  
      cout << " cognome: "; cin >> nuovoStudente.cognome;  
      cout << " nome: "; cin >> nuovoStudente.nome;  
      cout << " sesso( M/F ): "; cin >> nuovoStudente.sesso;  
      cout << " giorno di nascita (1-31): ";  
      cin >> nuovoStudente.dataNascita.giorno;  
      cout << " mese di nascita (1-12): ";
```

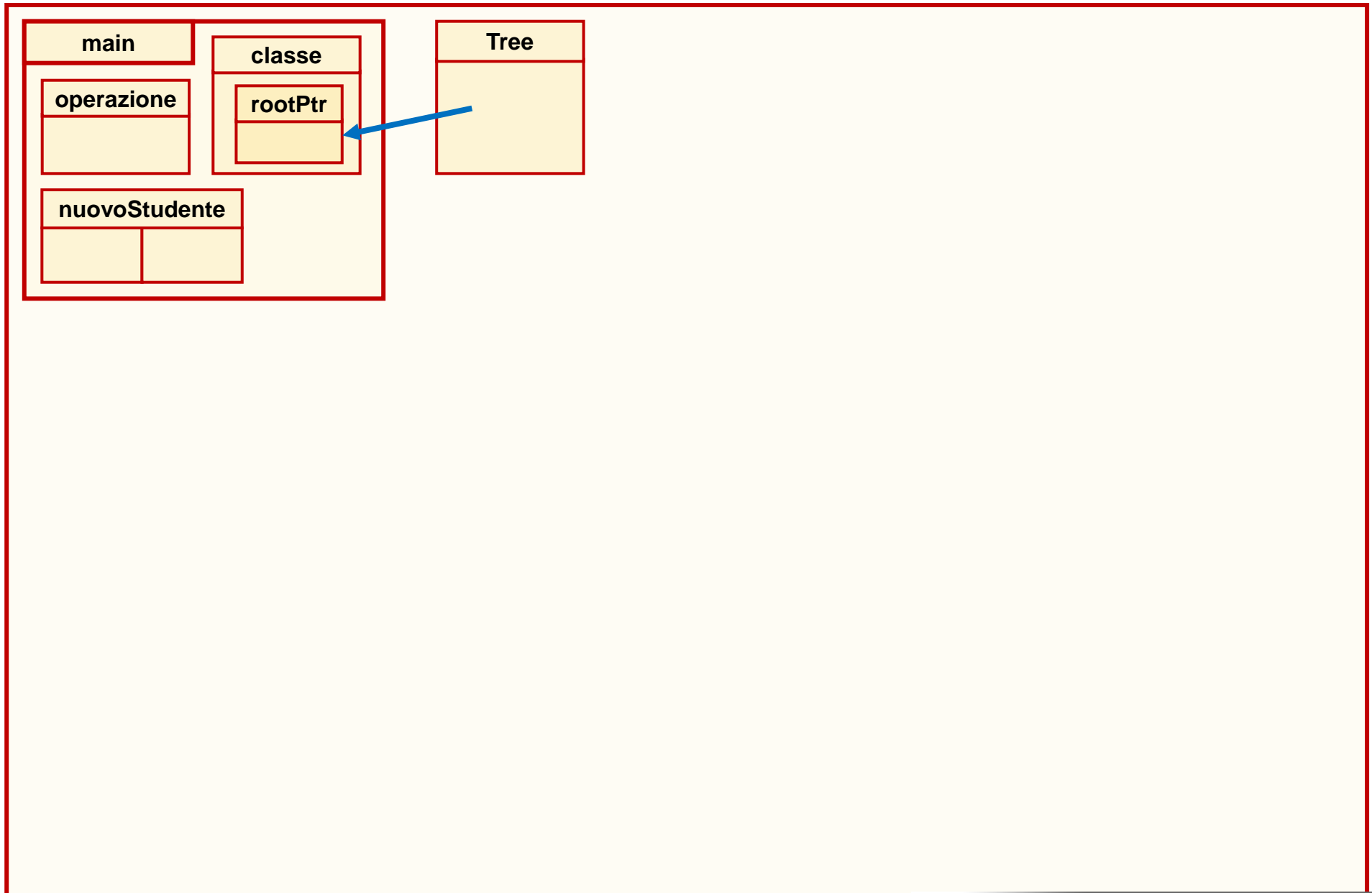
Modello ad ambienti

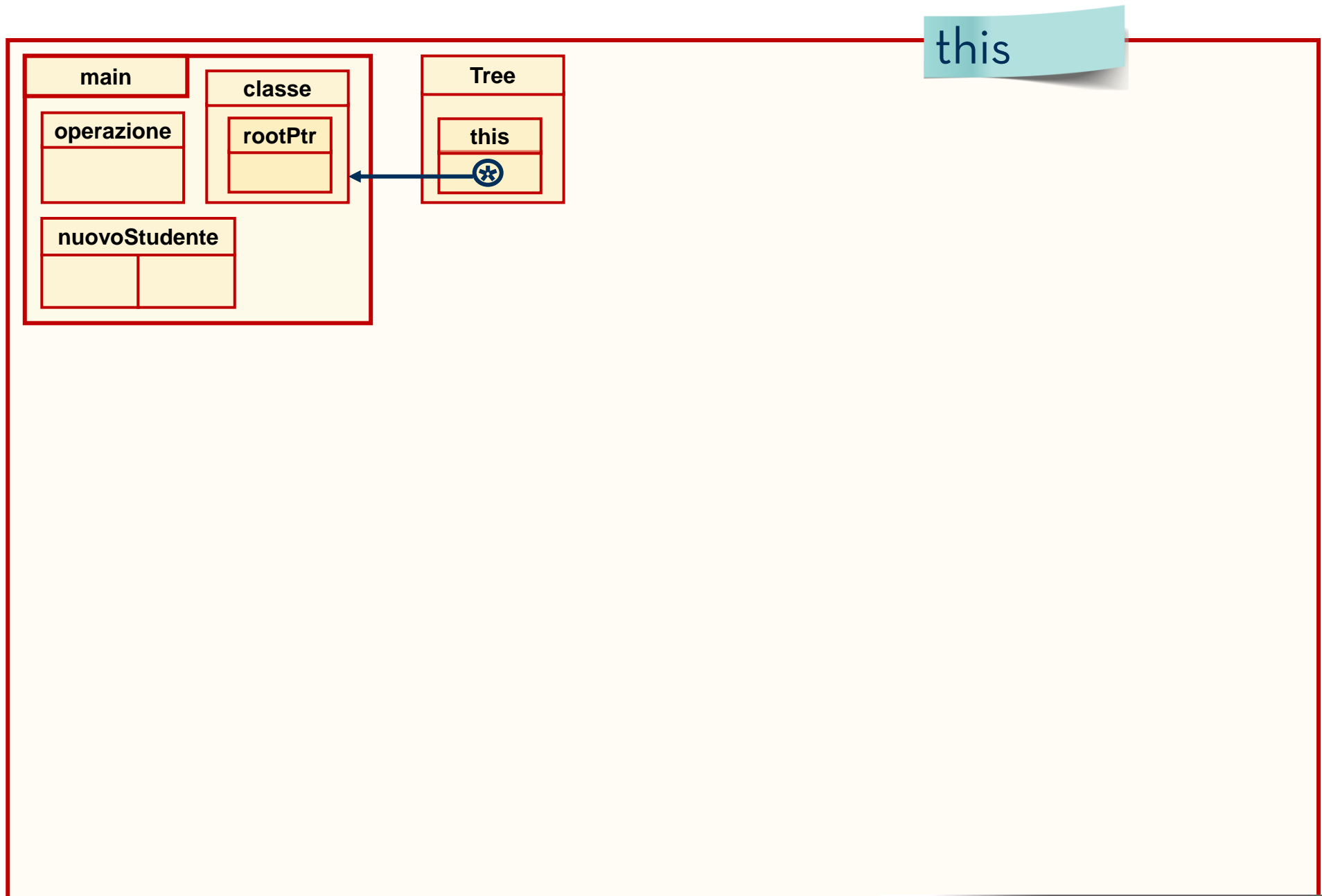


nuovoStudente									
matricola									
cognome									
nome									
sex									
dataNascita	giorno								
	mese								
	anno								

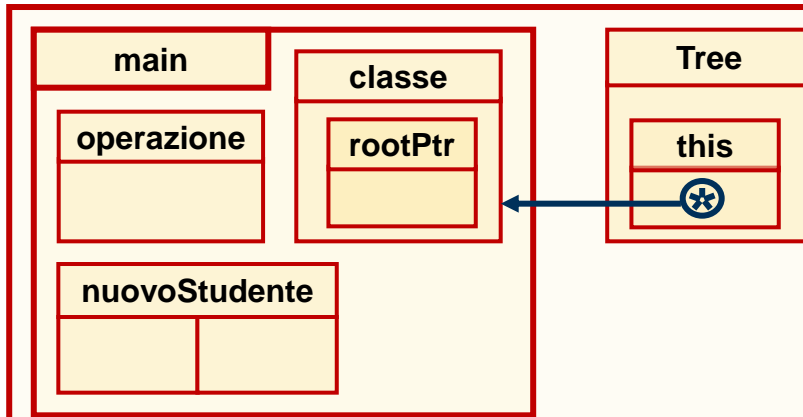




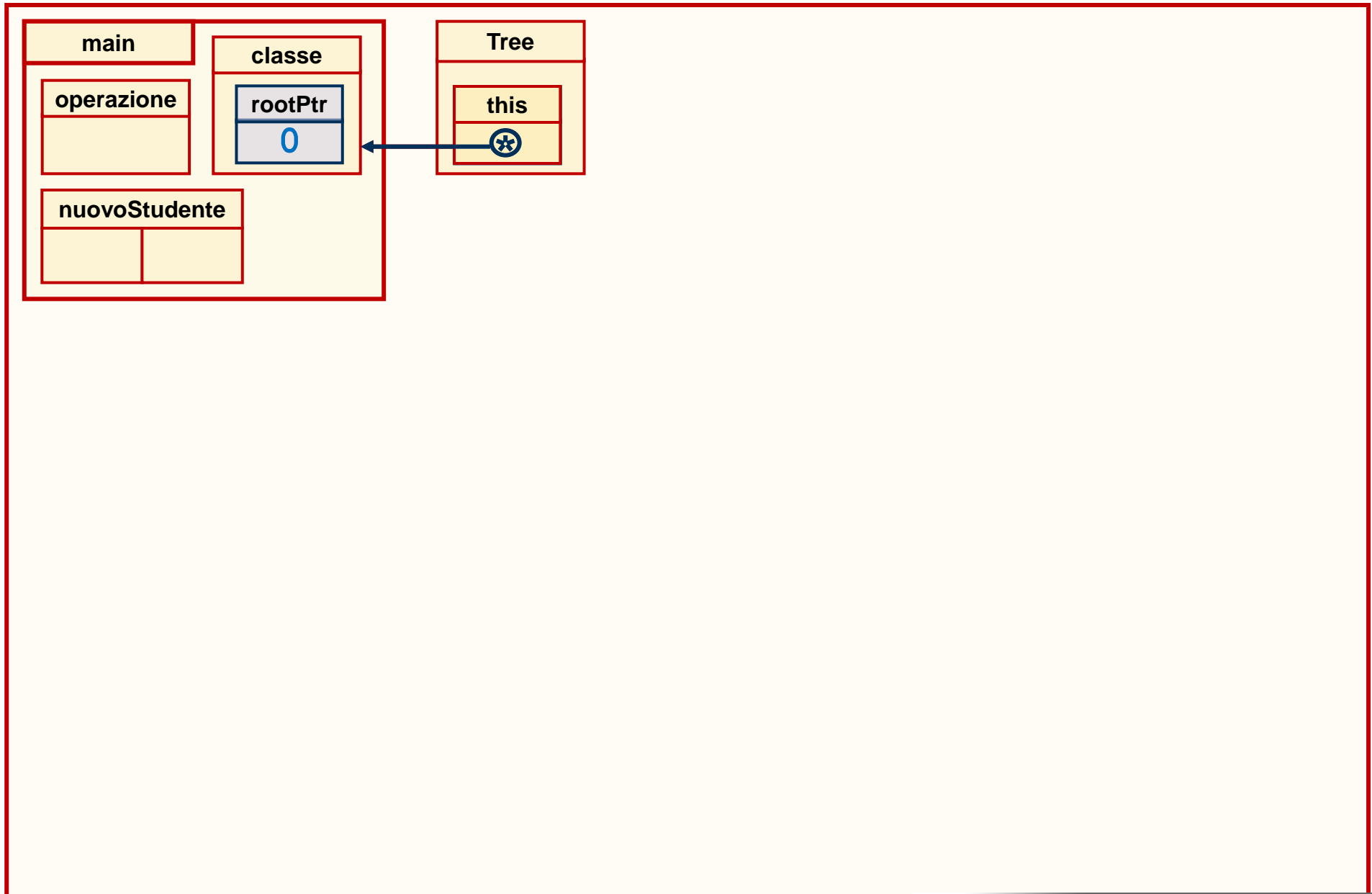


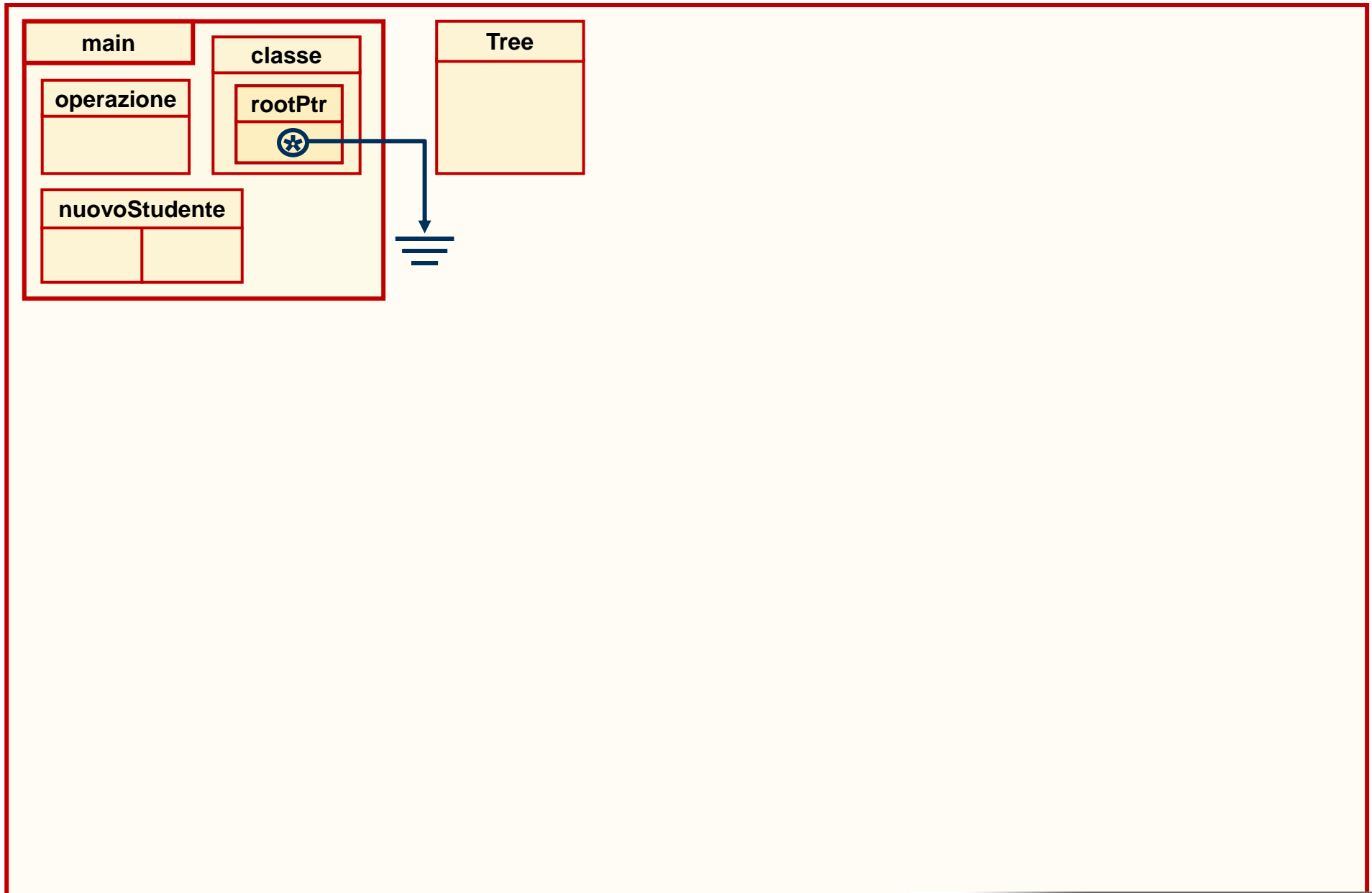


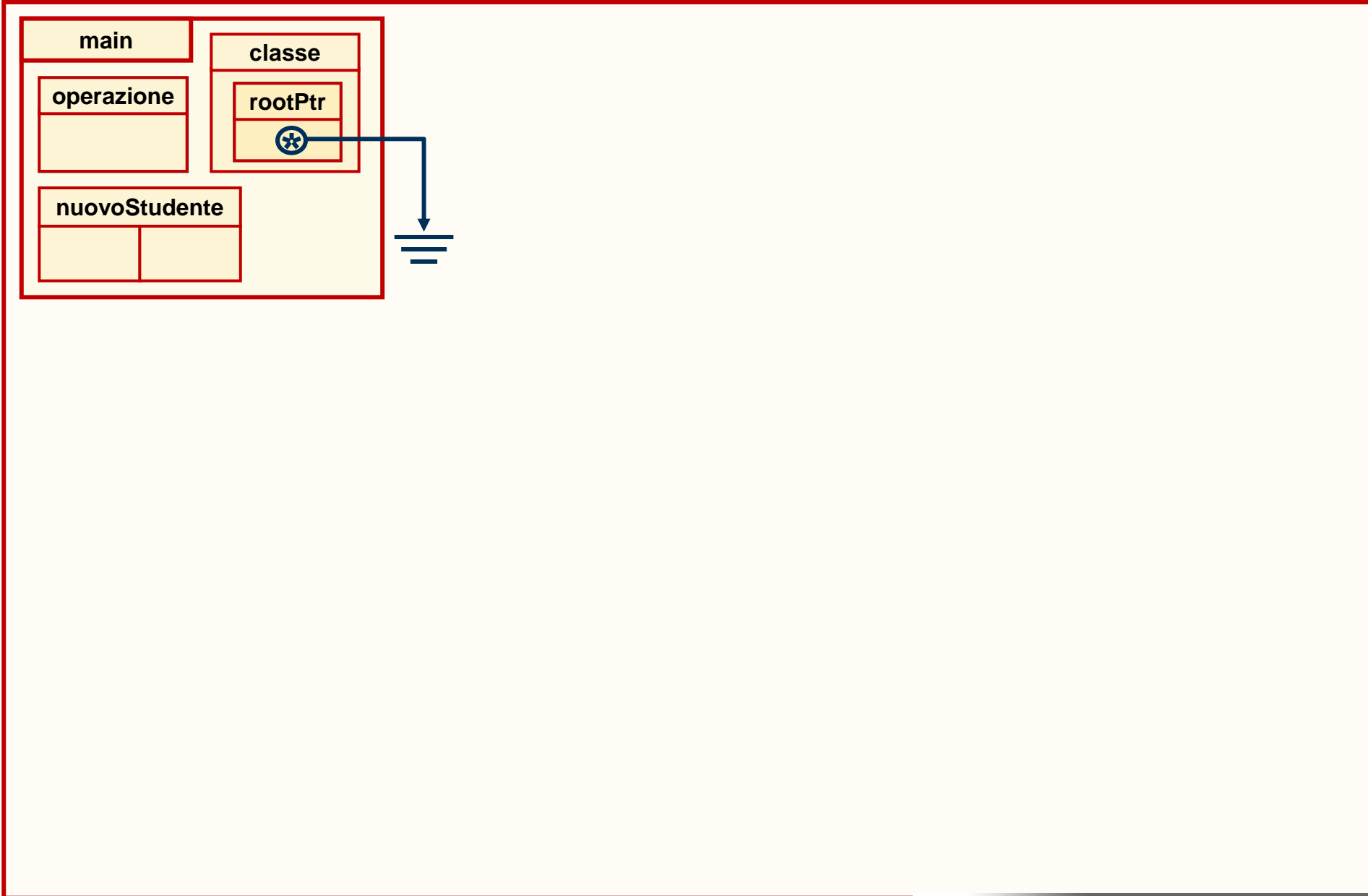
Parametro implicito

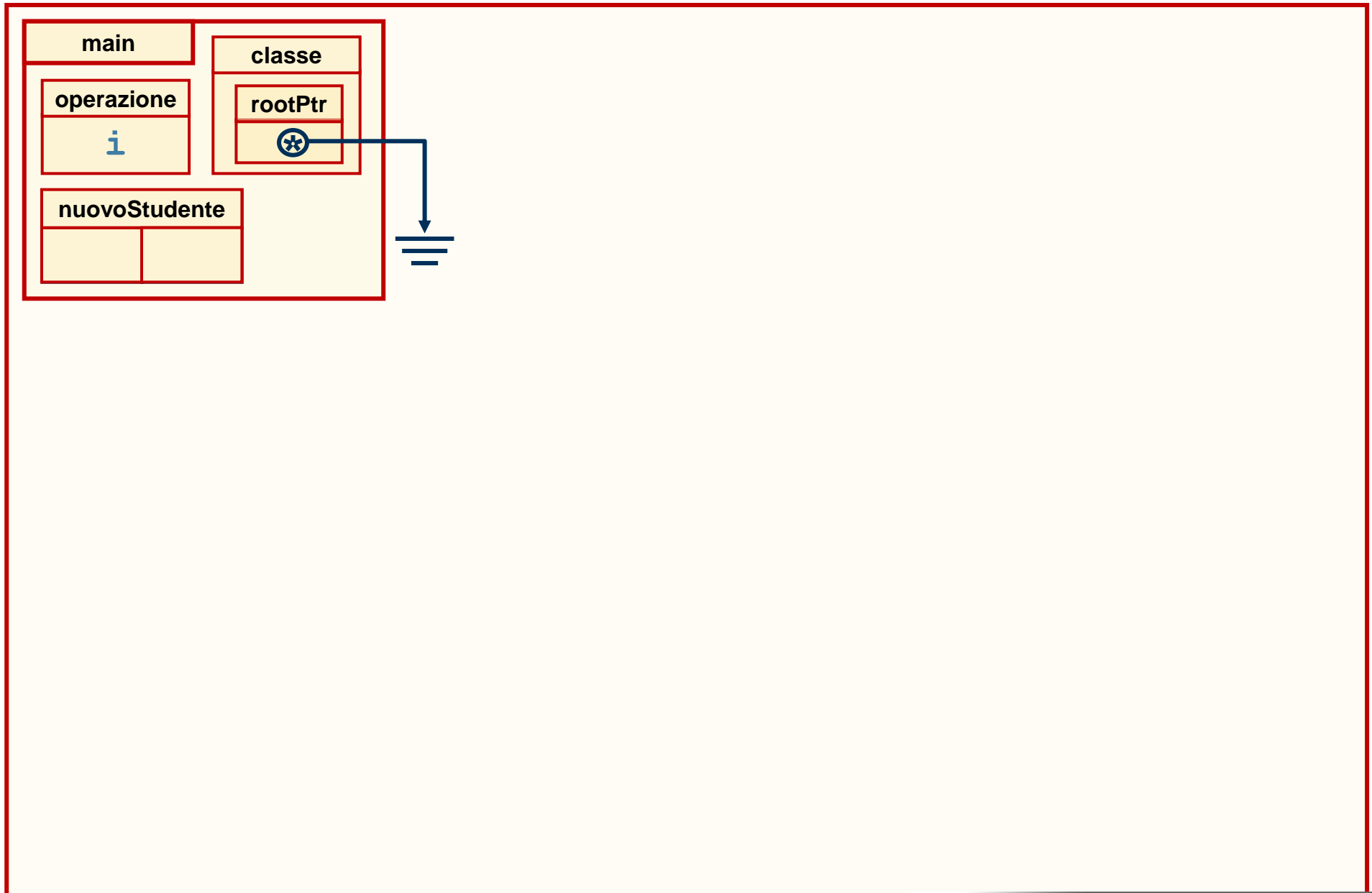


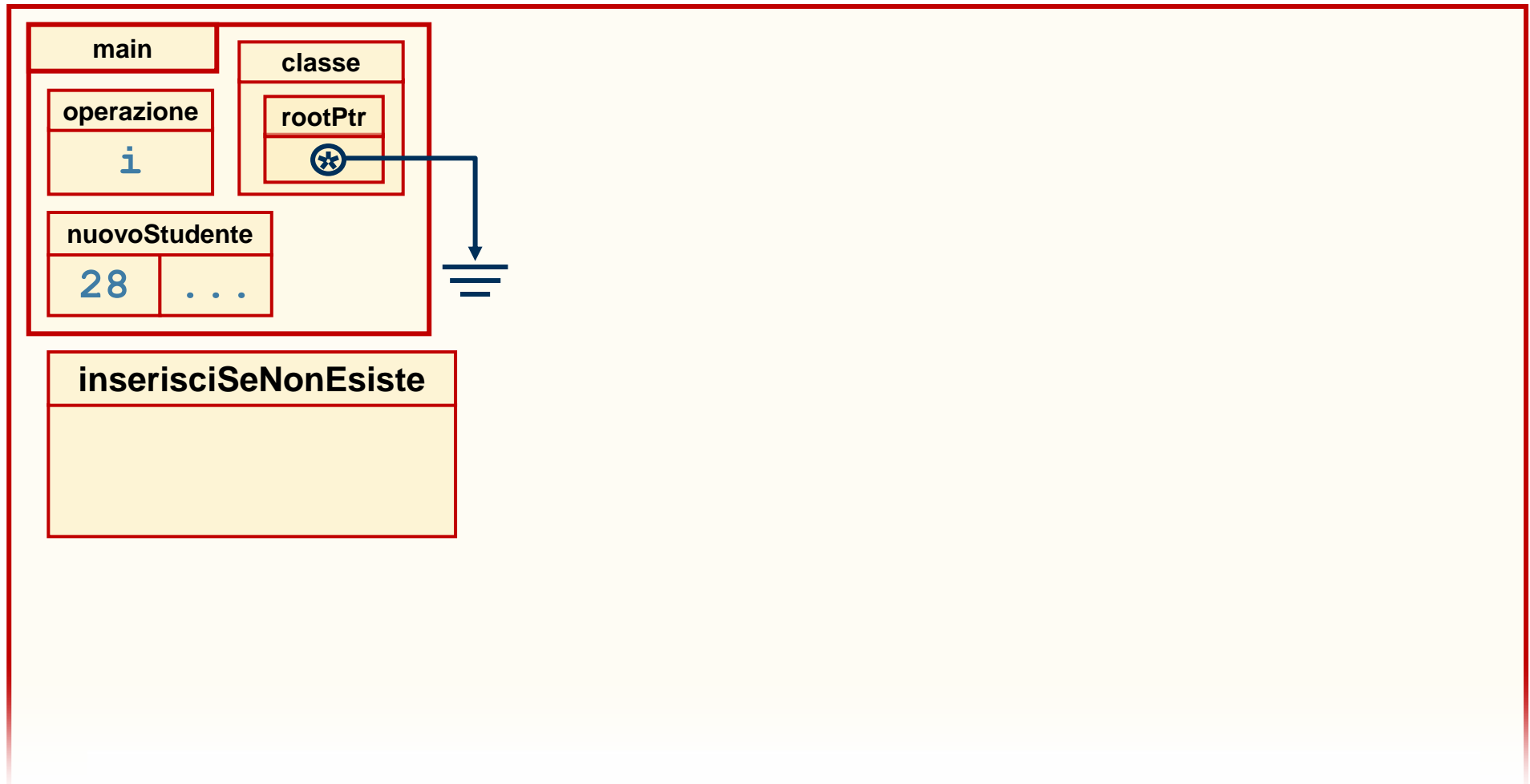




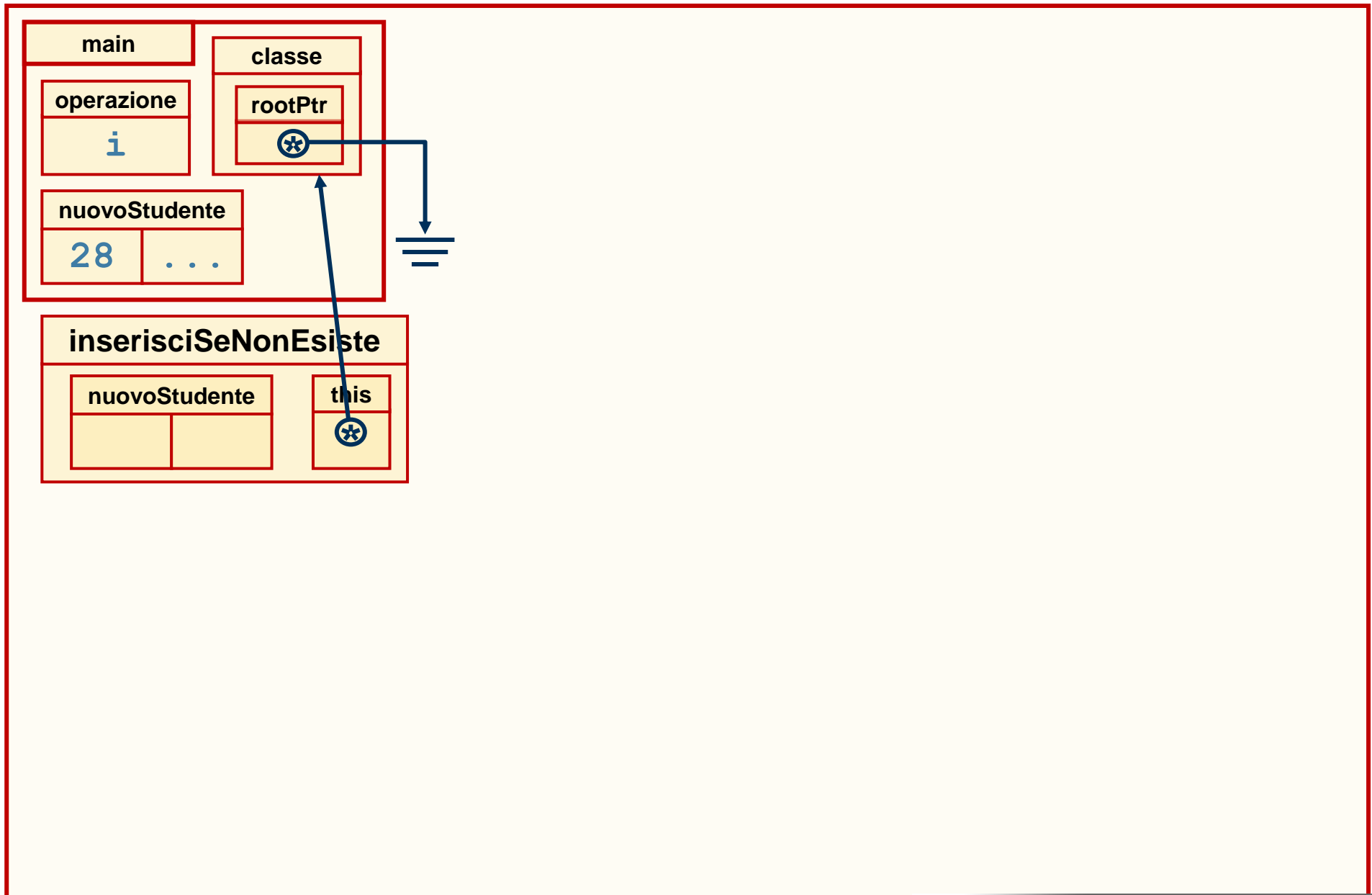


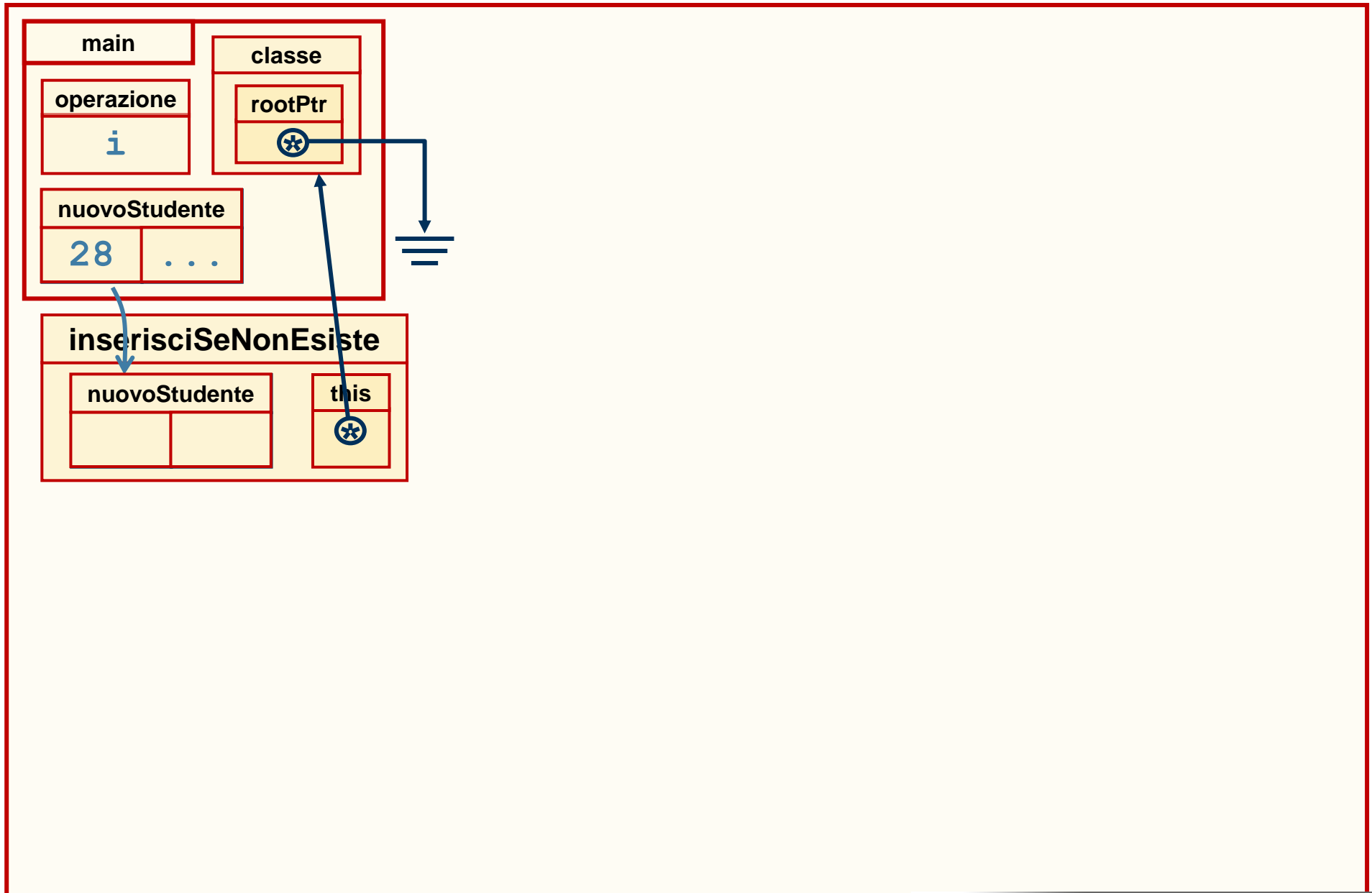




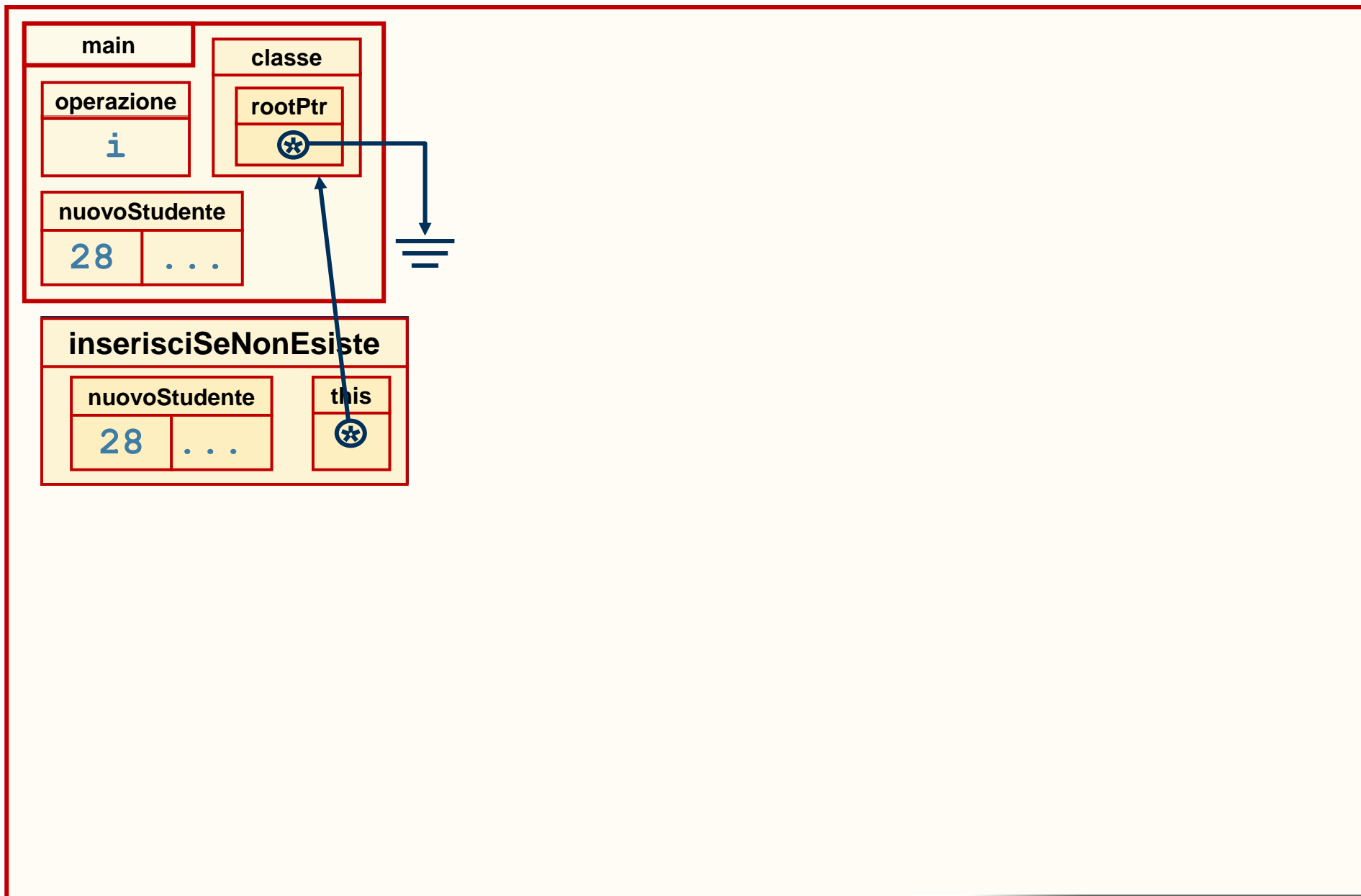


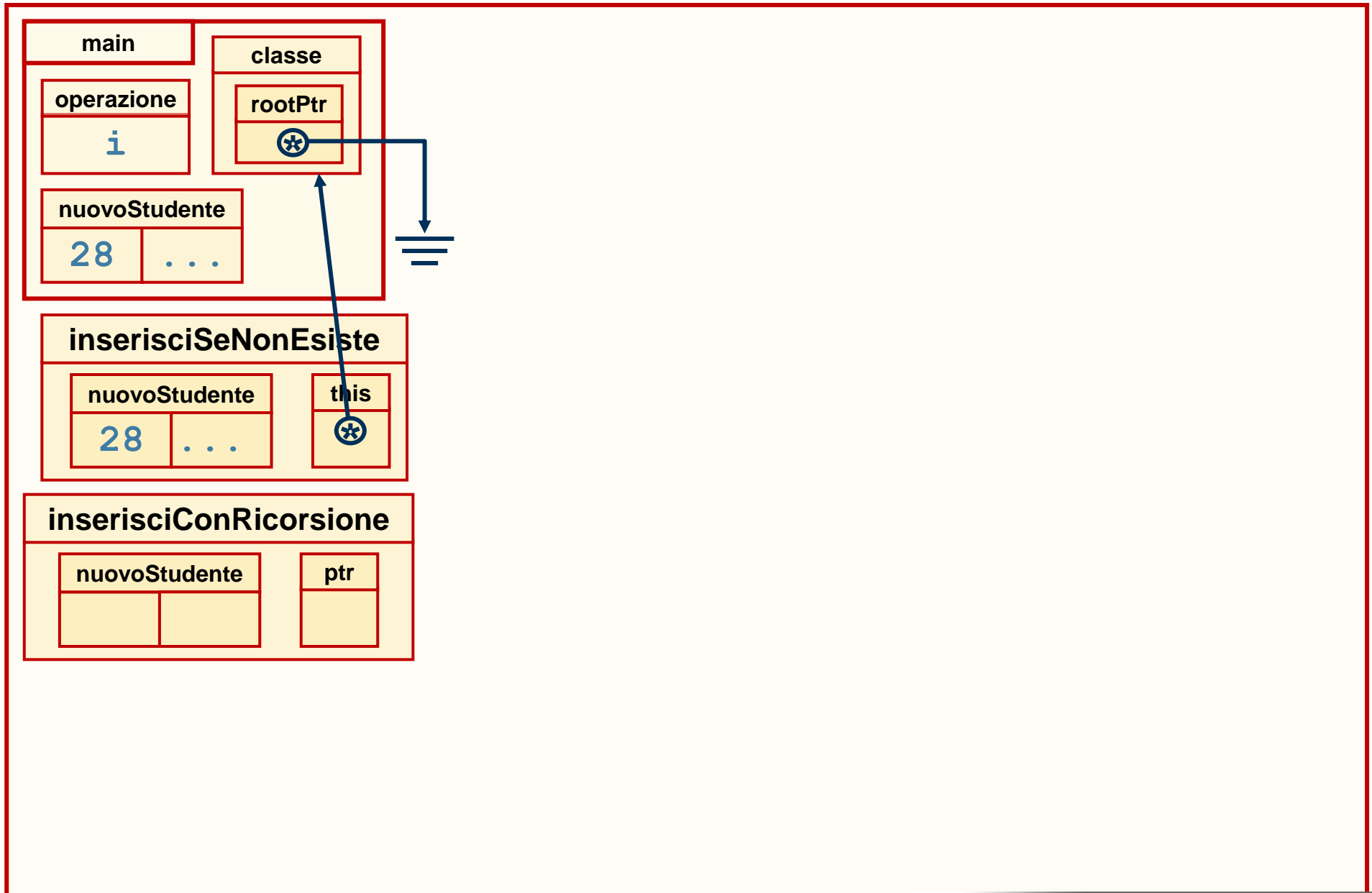
```
cout << " anno di nascita (0-99): ";  
cin >> nuovoStudente.dataNascita.anno;  
classe.inserisciSeNonEsiste(nuovoStudente);  
}  
cout << "operazione? ( i per inserimento,"  
    << " e per eliminazione, f per fine" << " )";
```



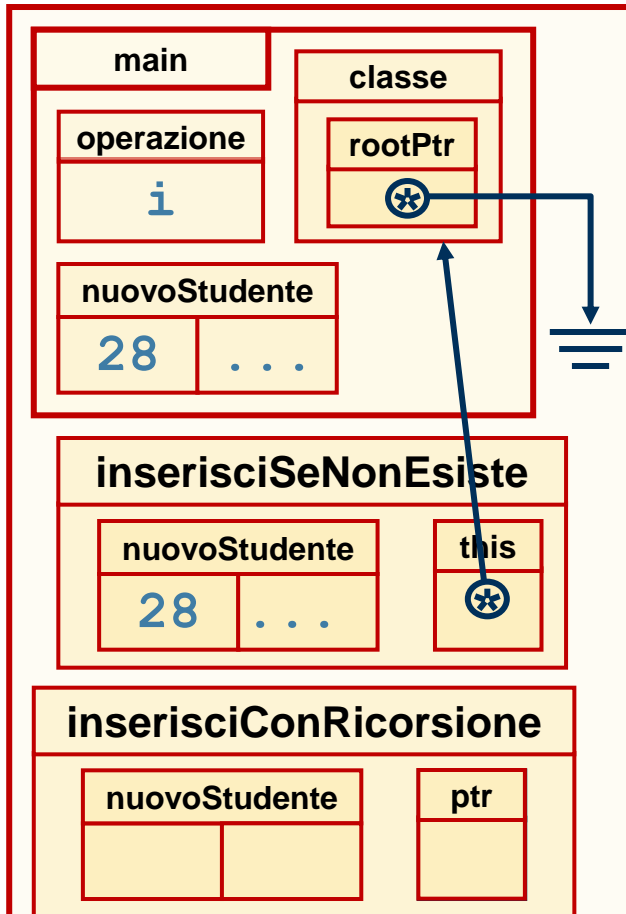




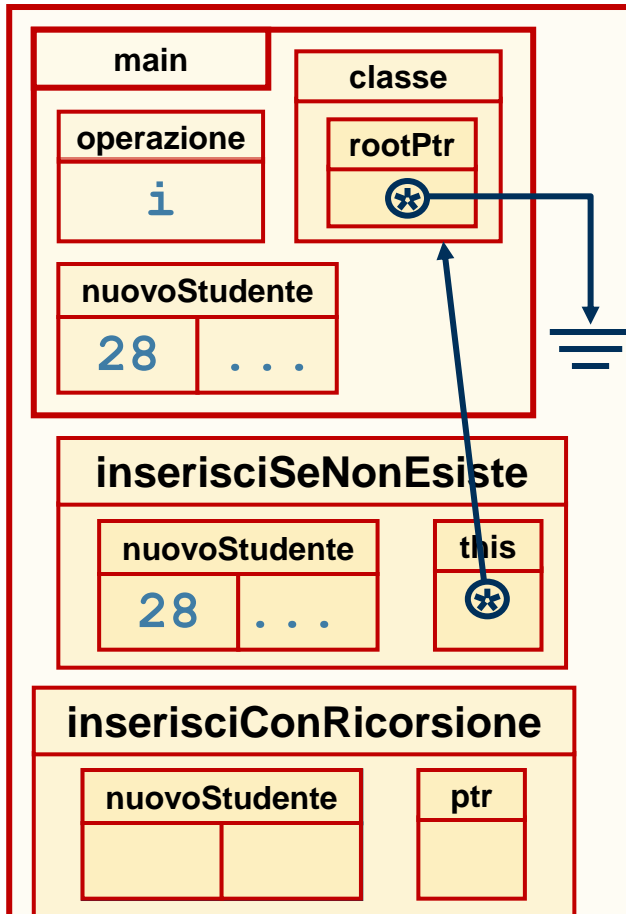


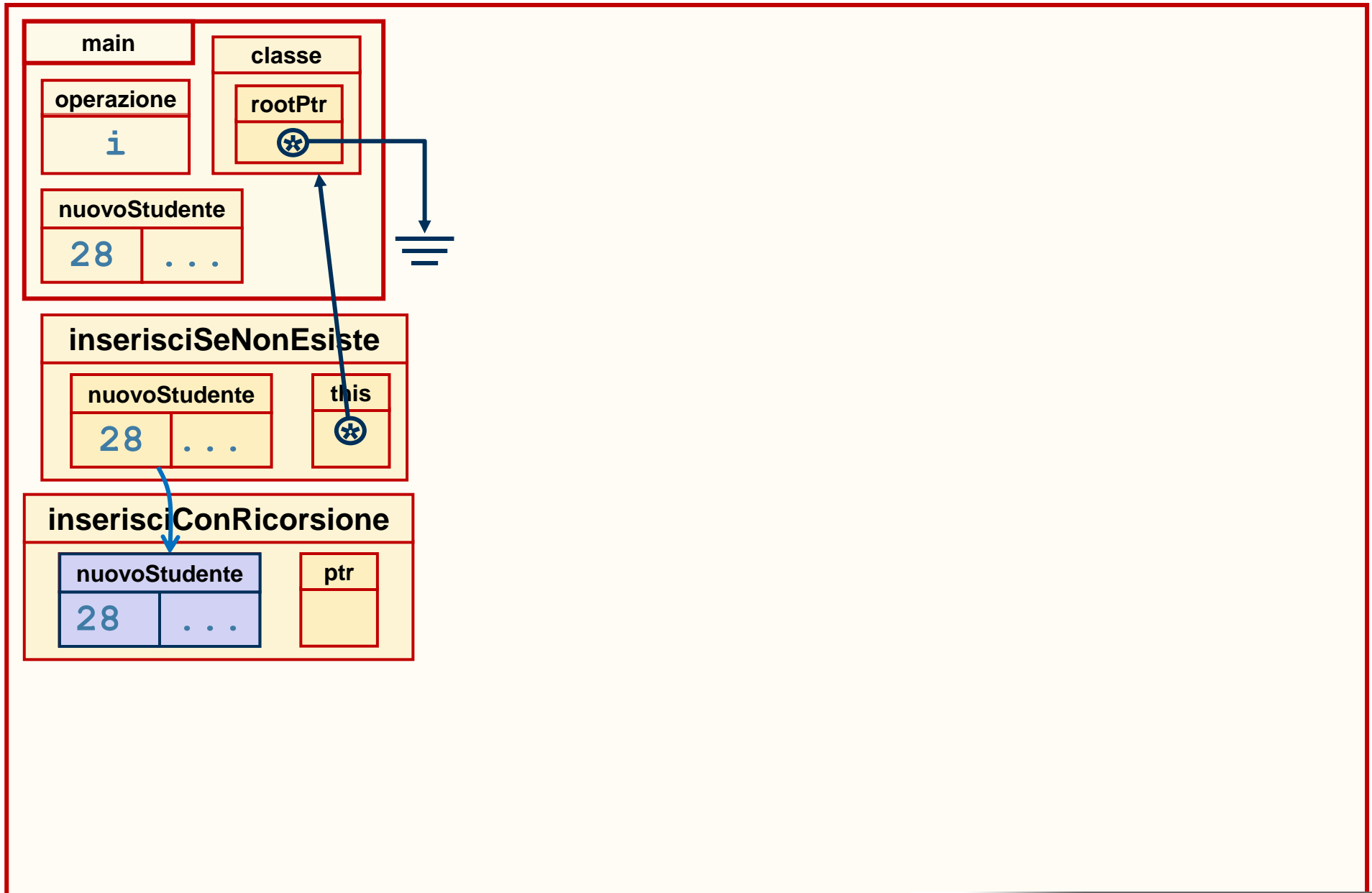


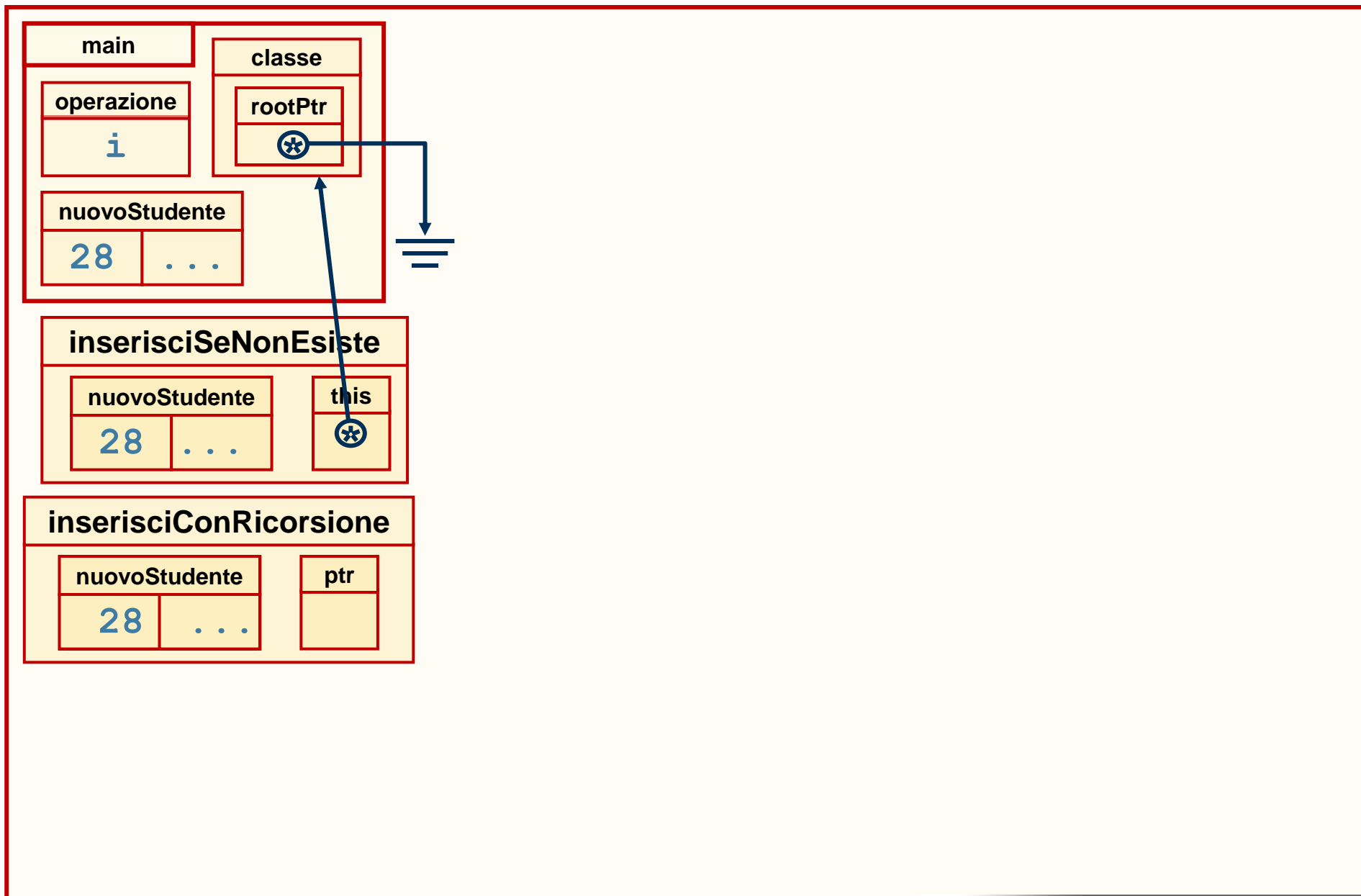
Per riferimento

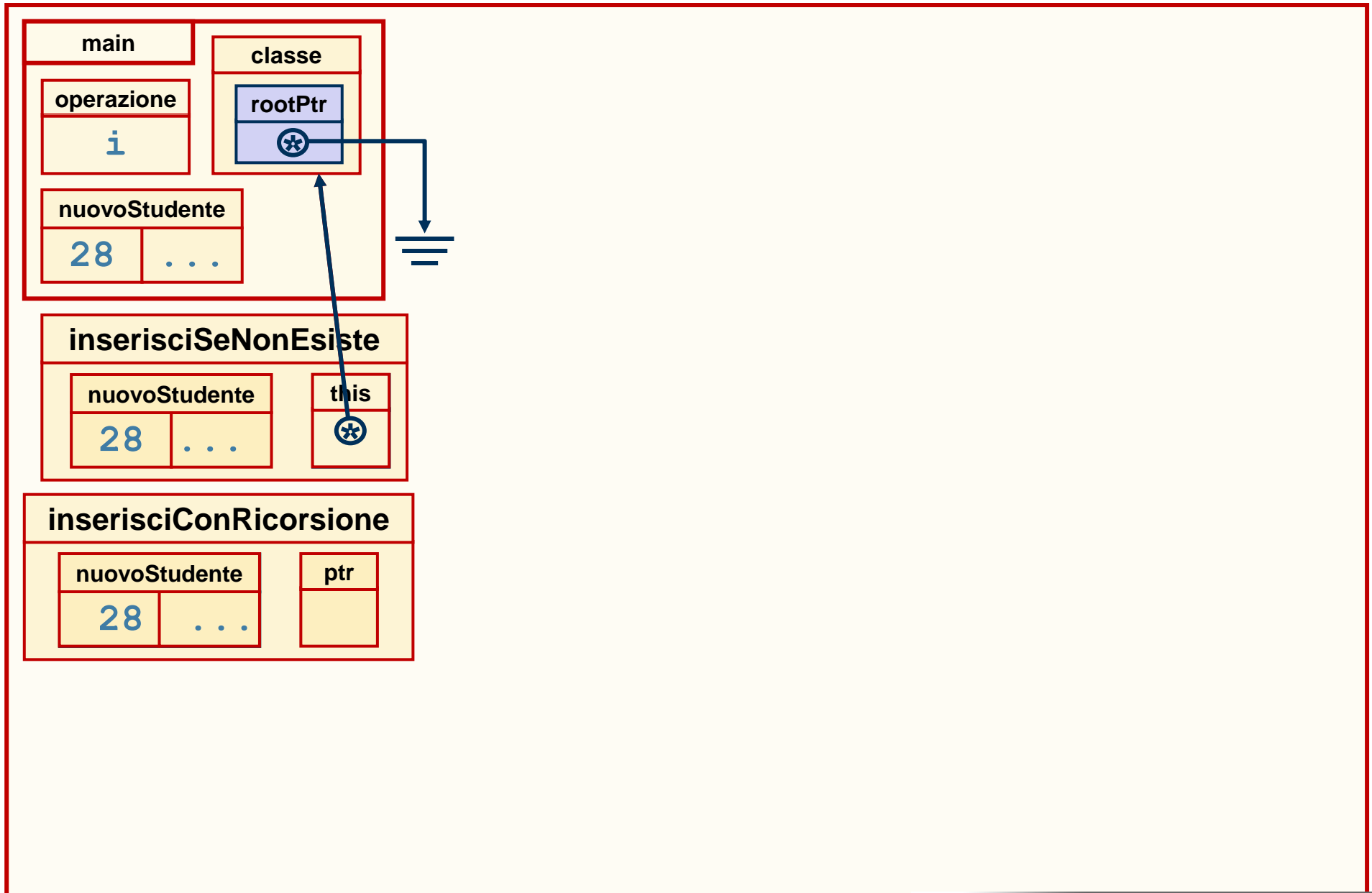


Per valore

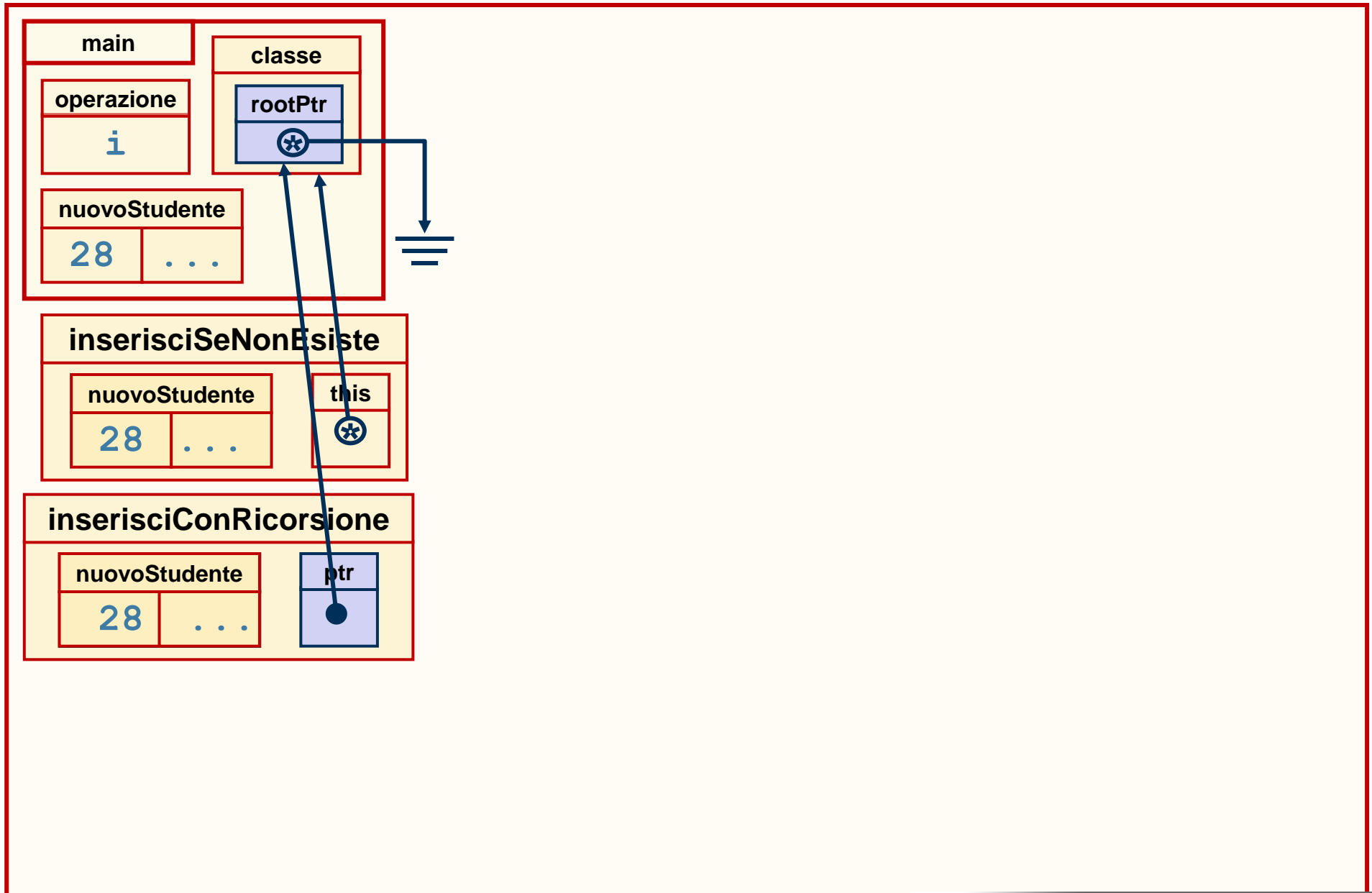


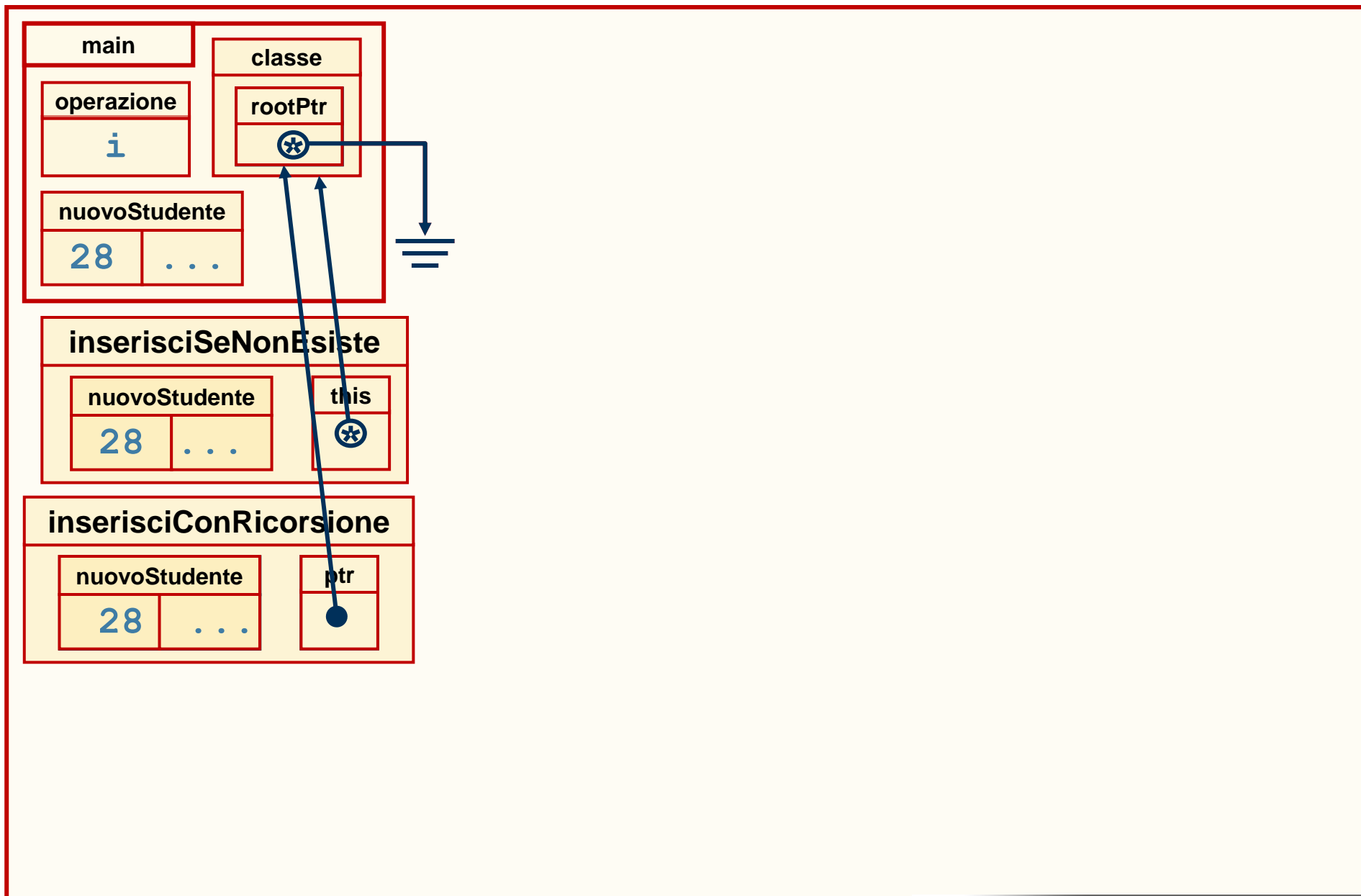


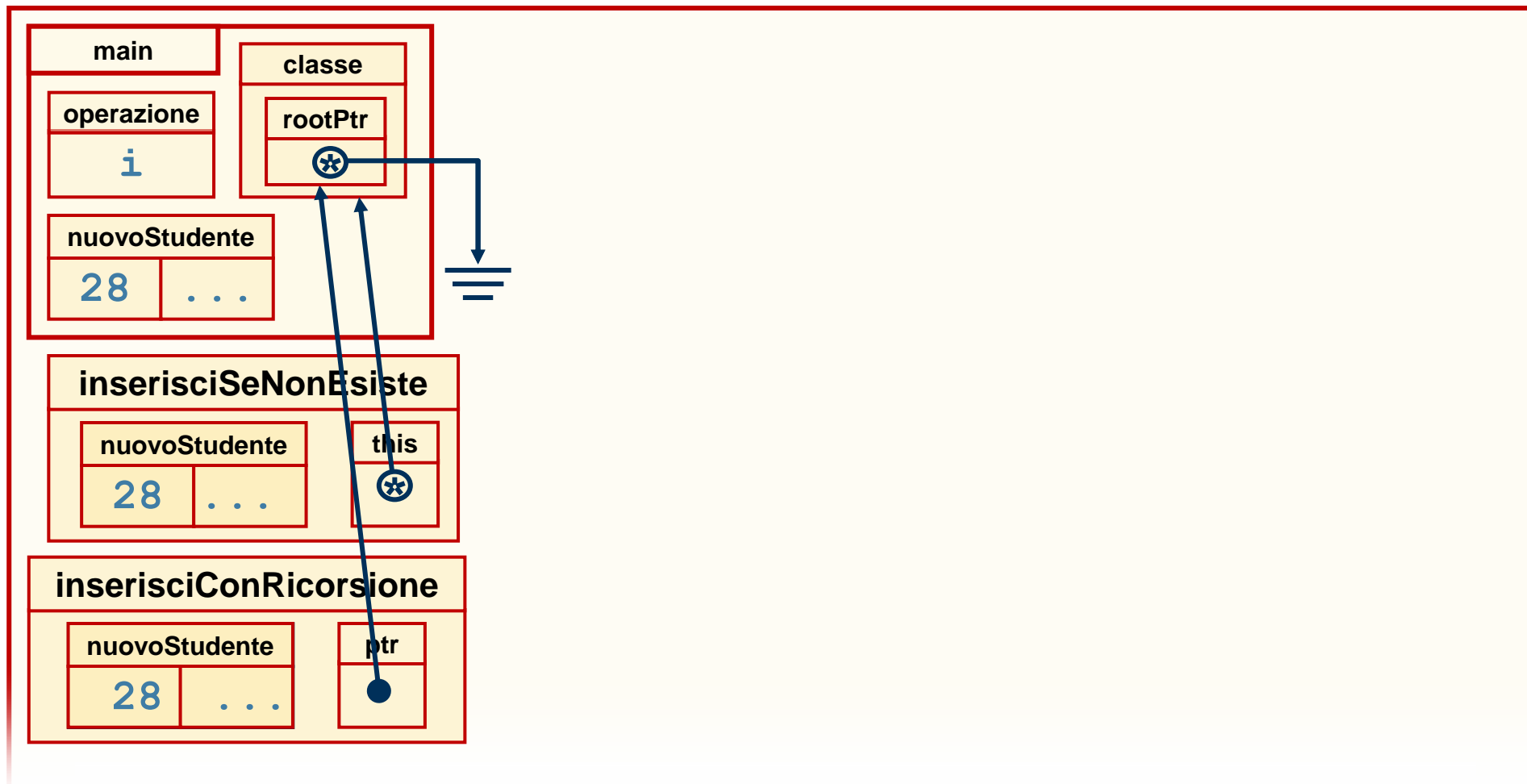






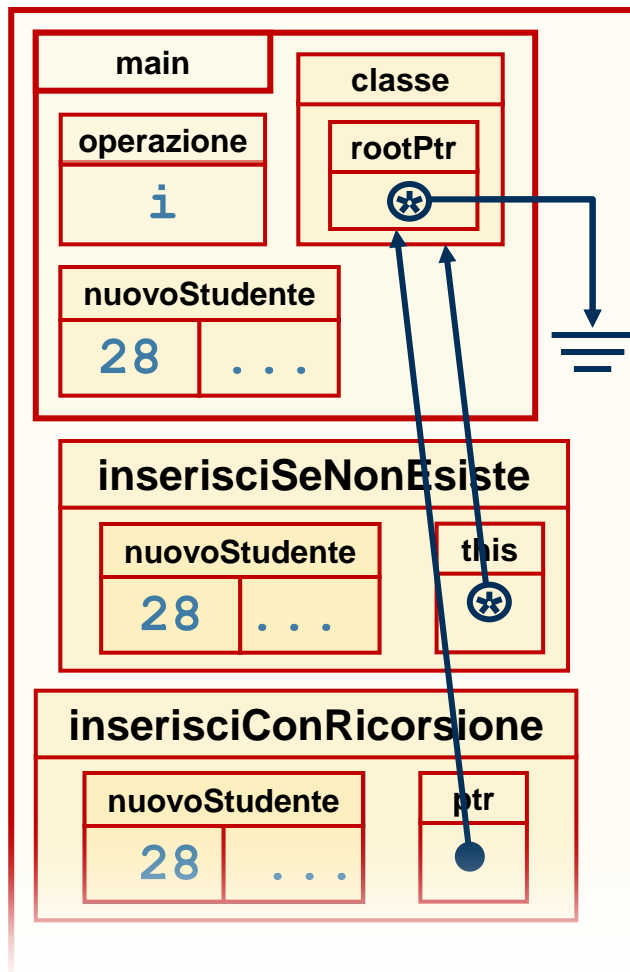




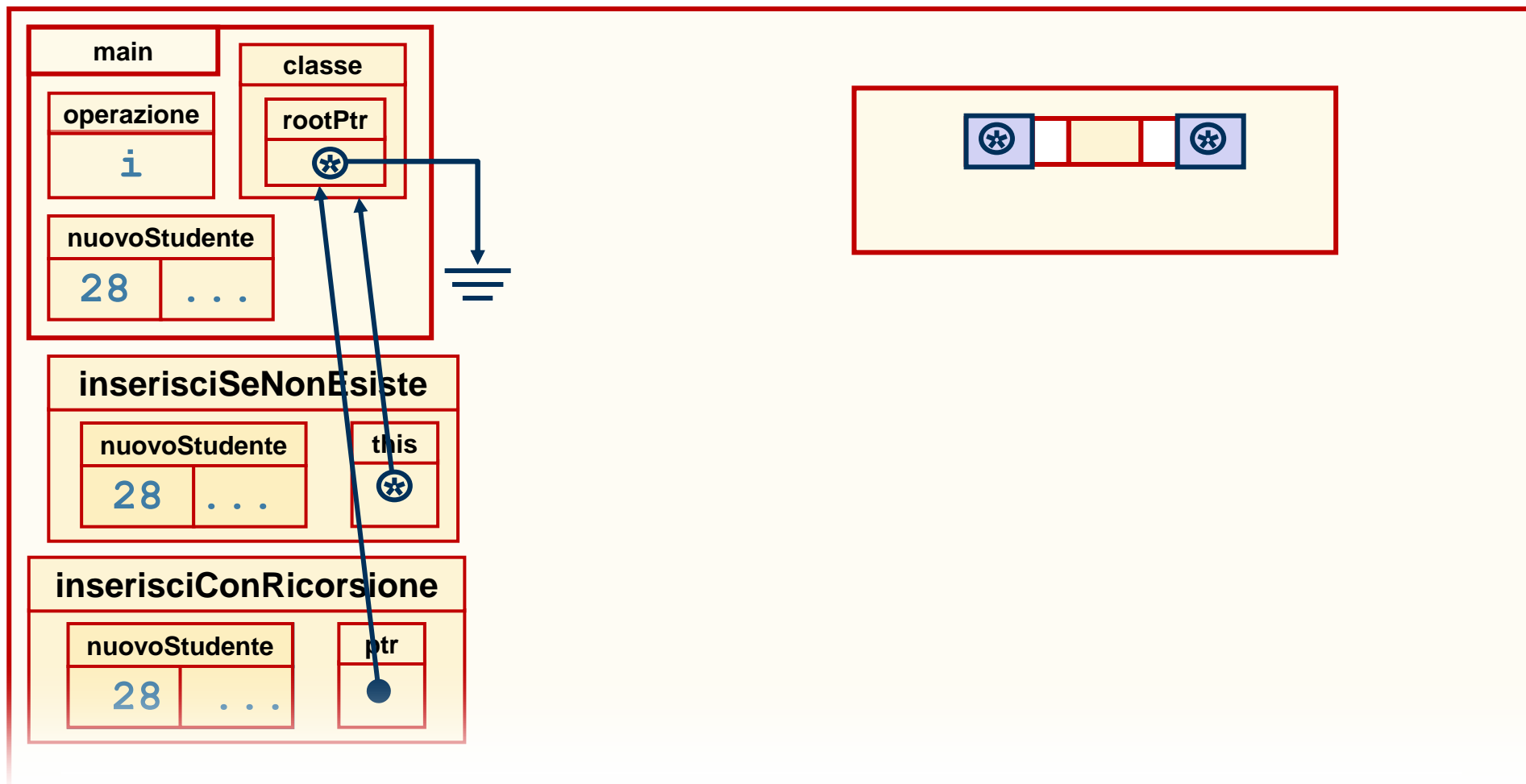


```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```

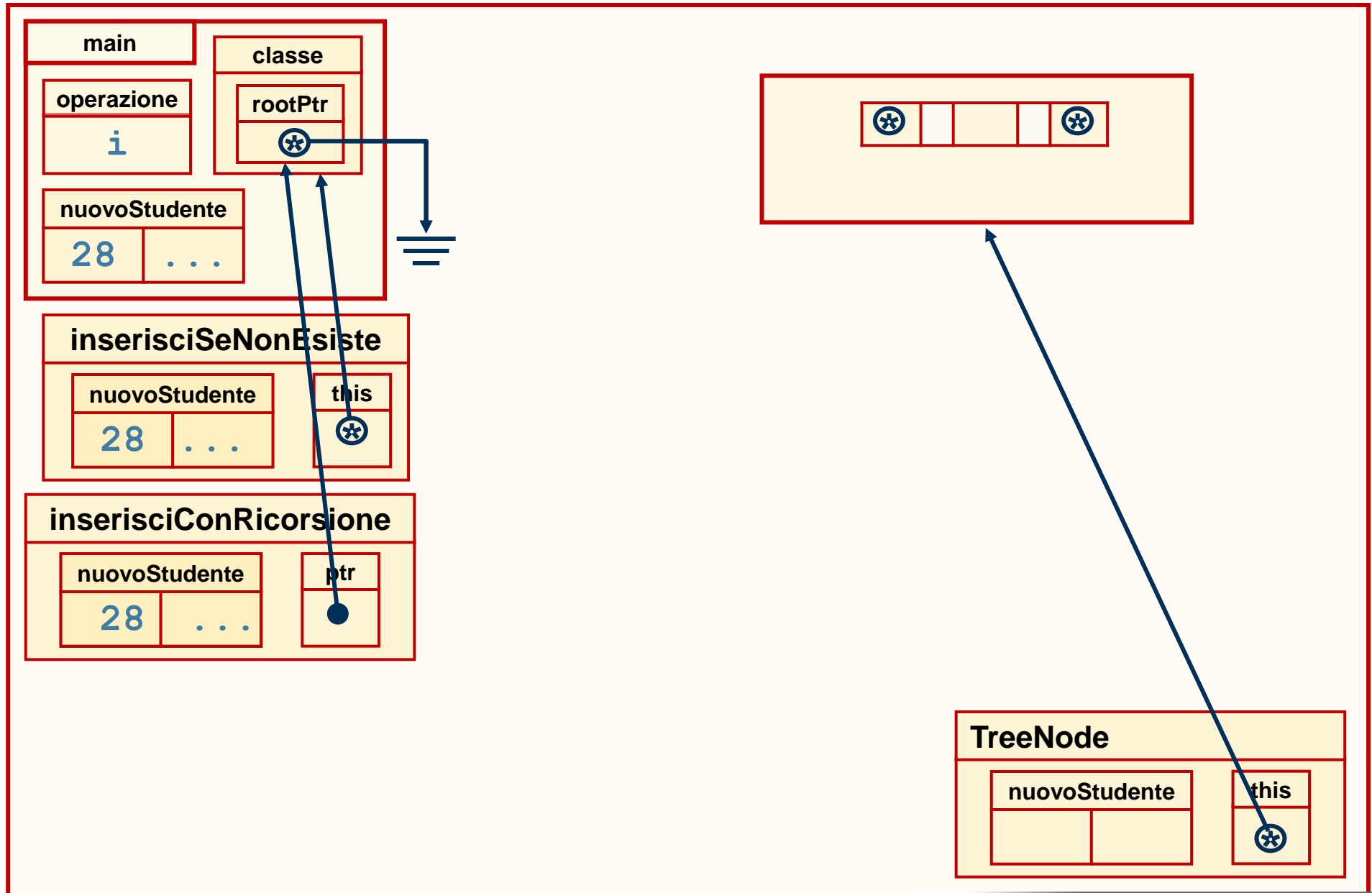
## Assegnazione

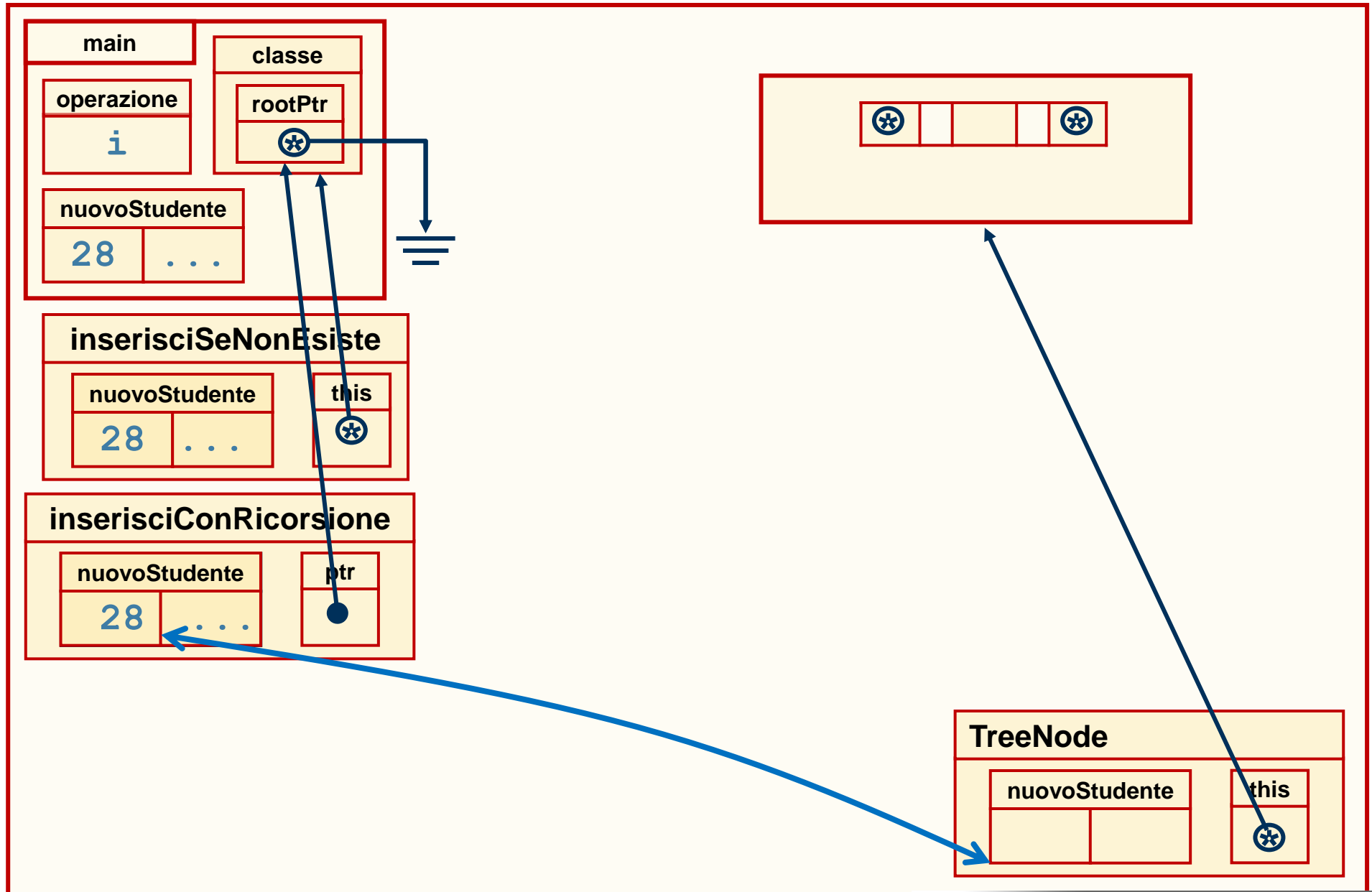


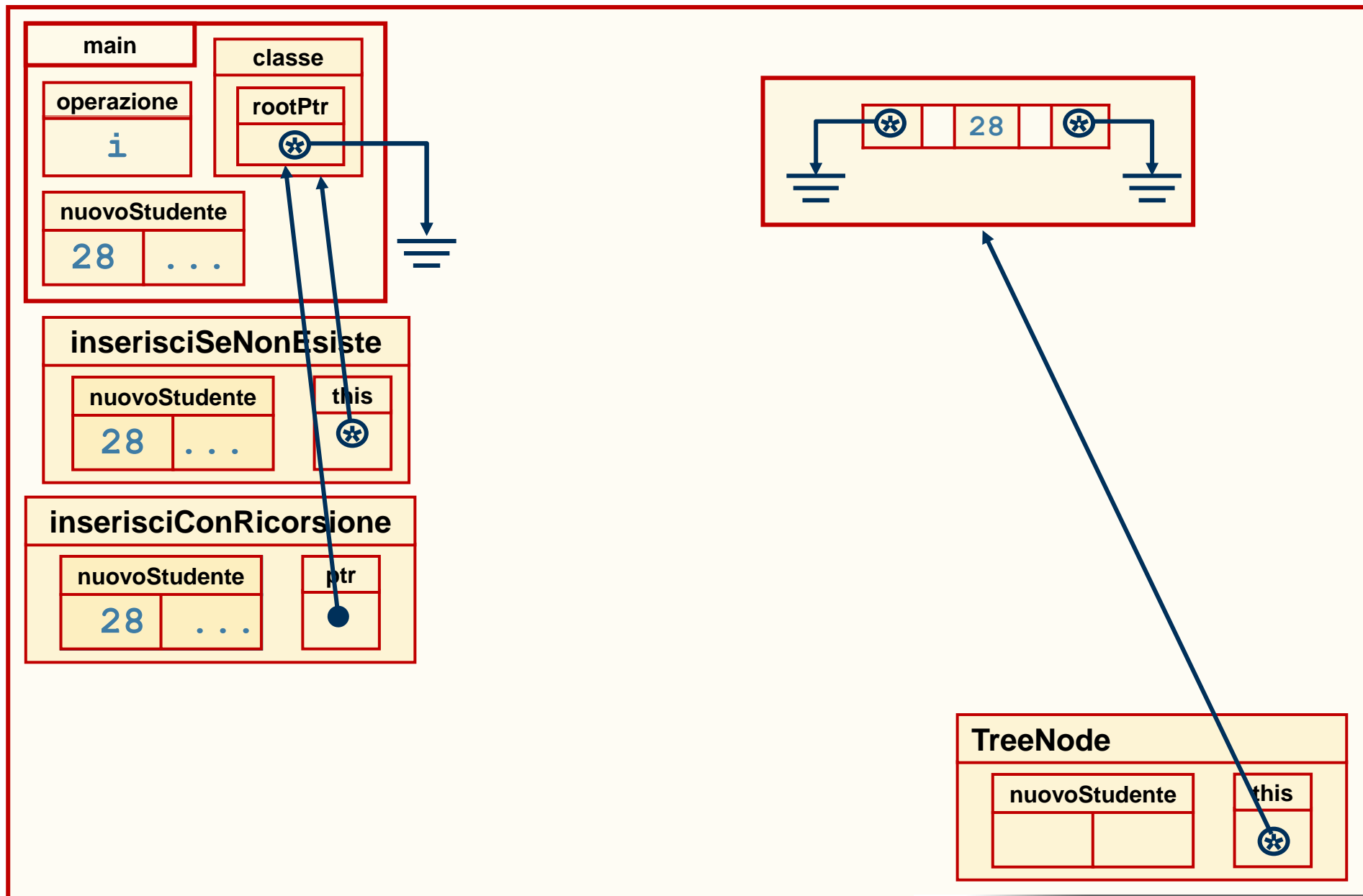
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```



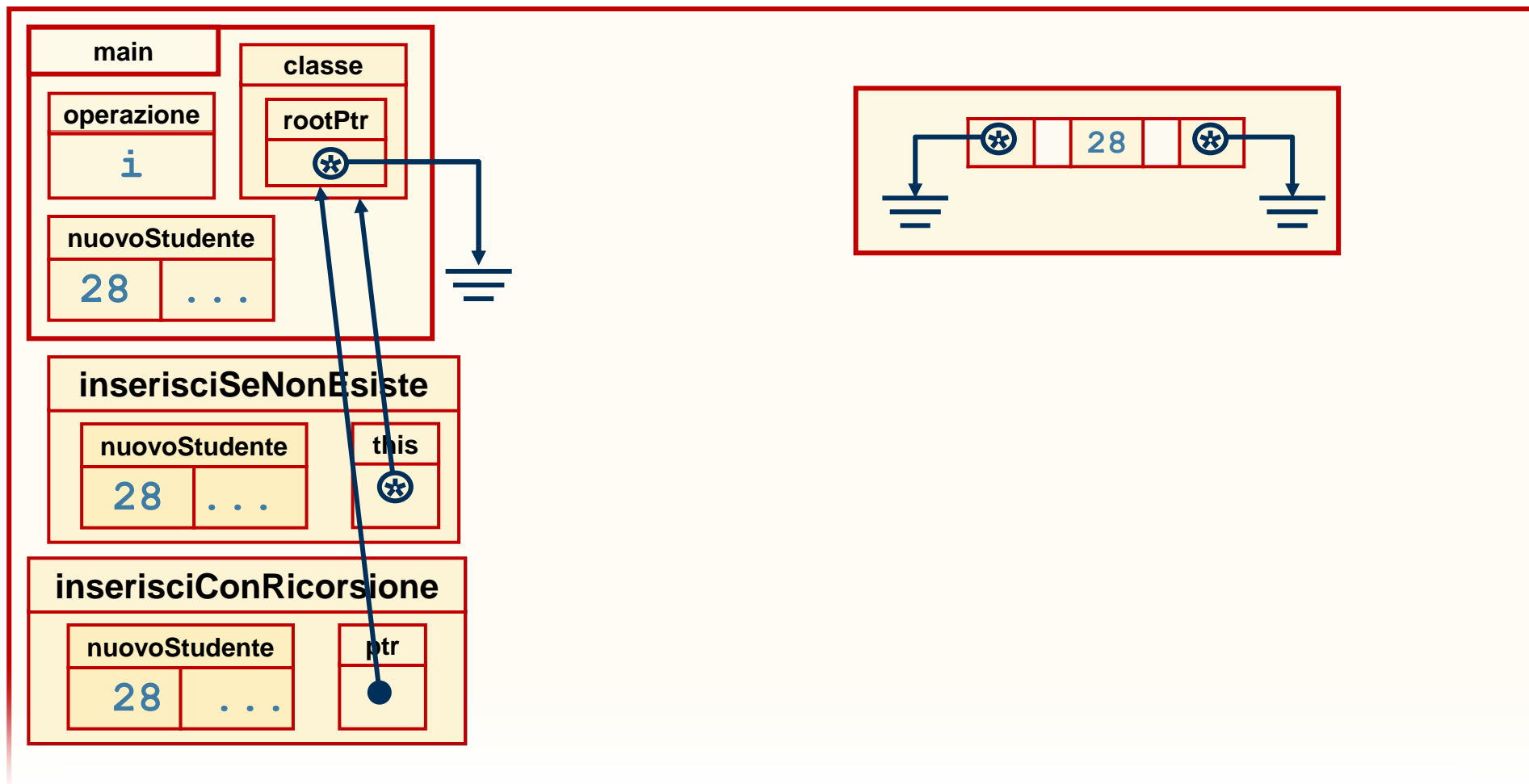
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```



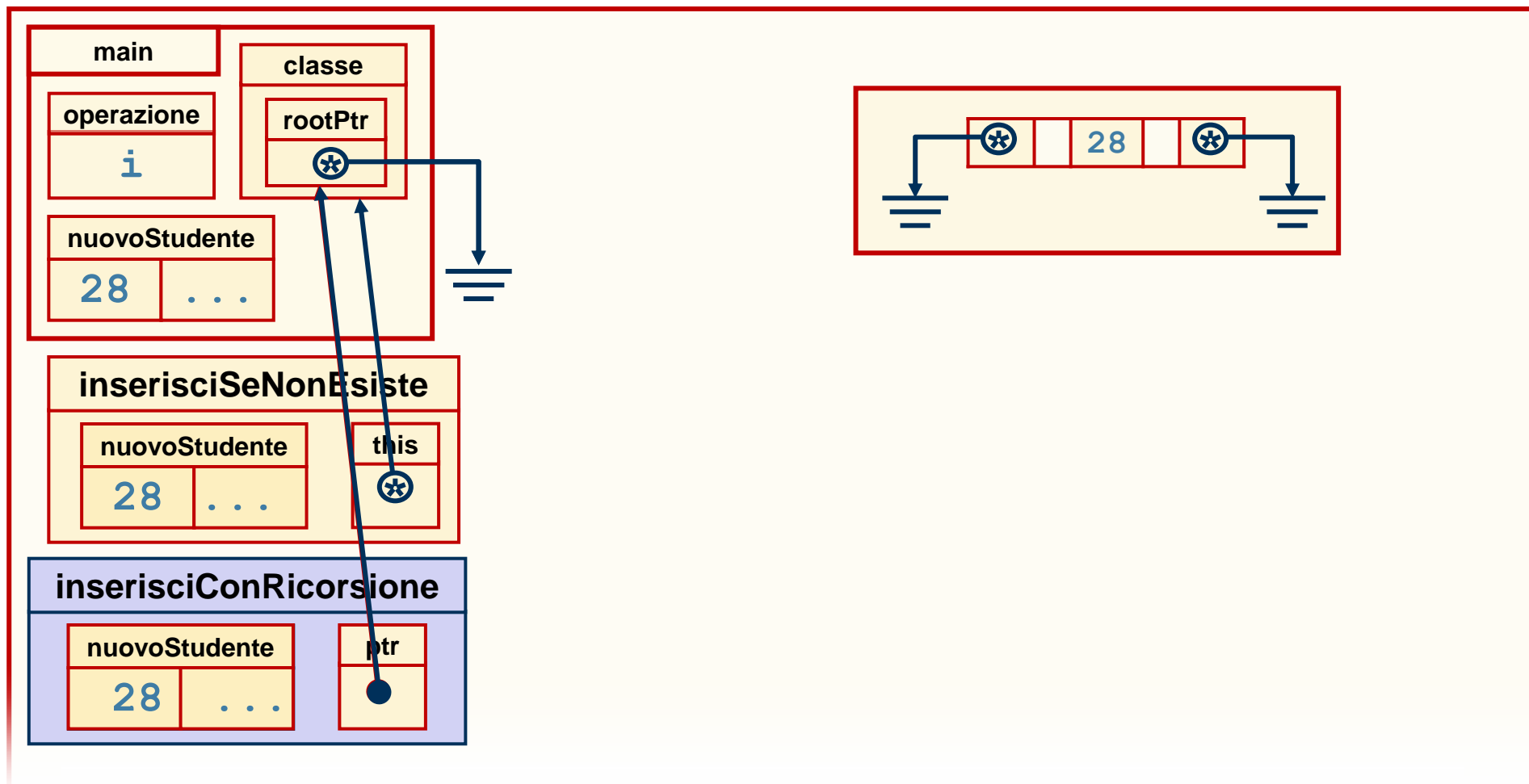




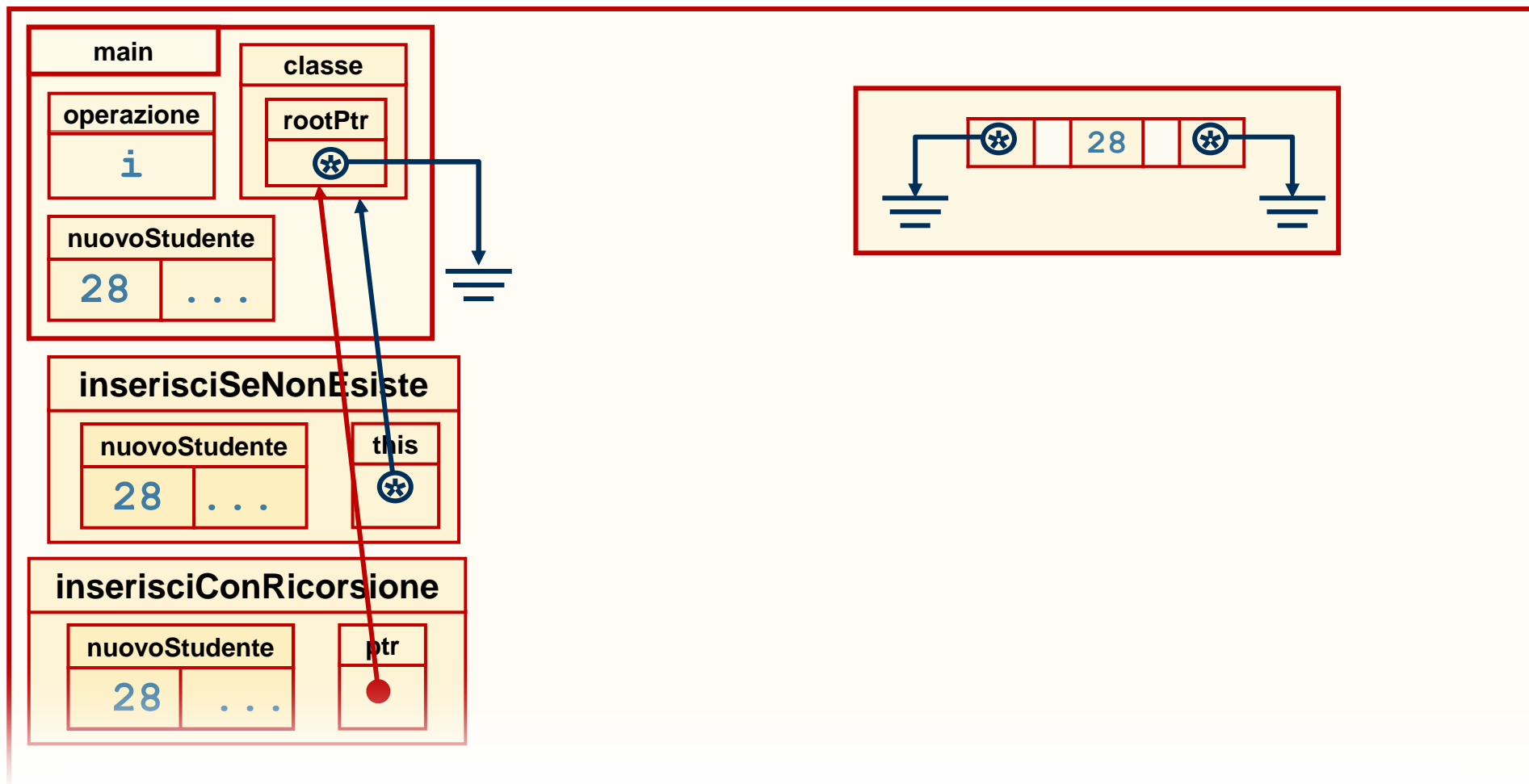




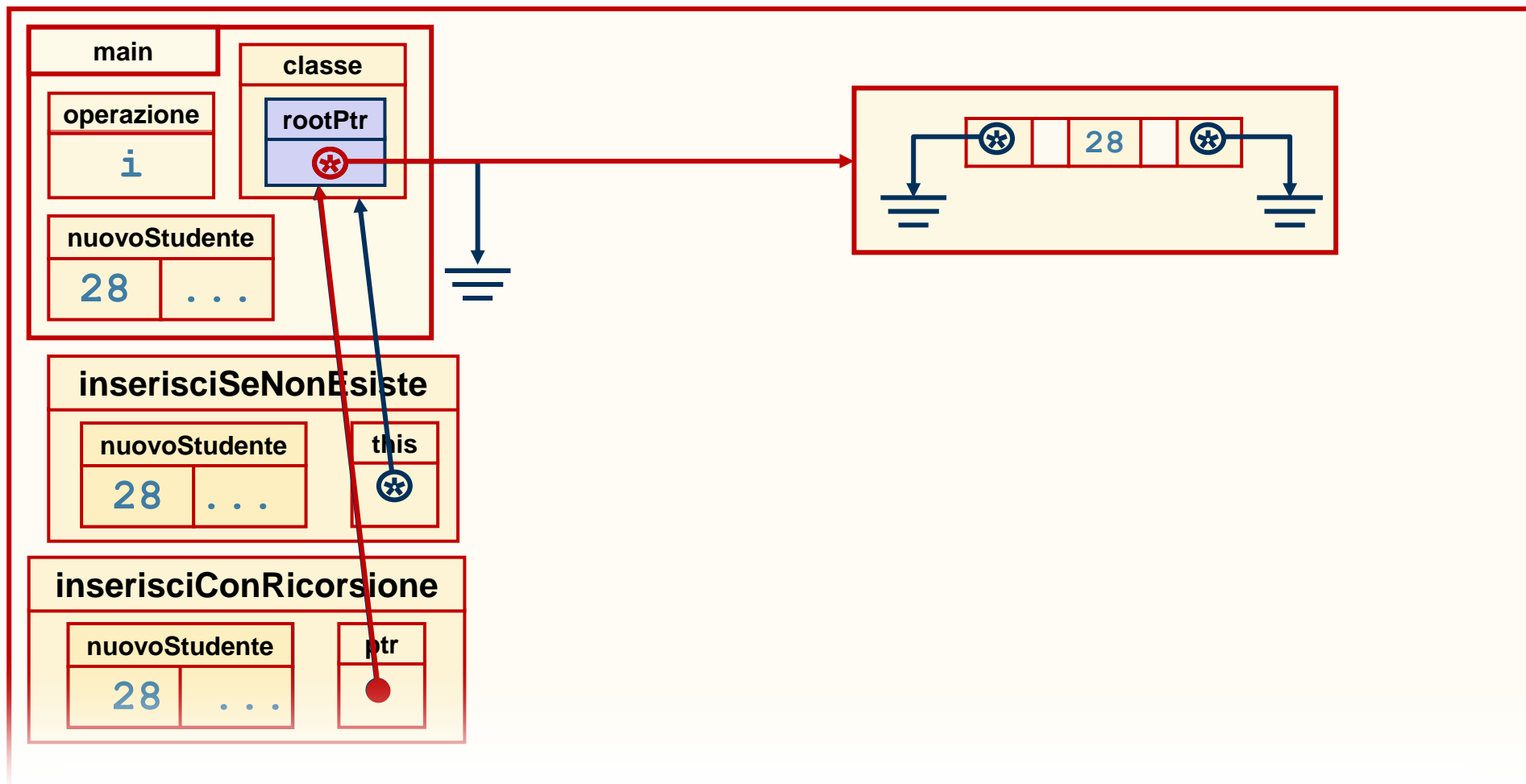
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```



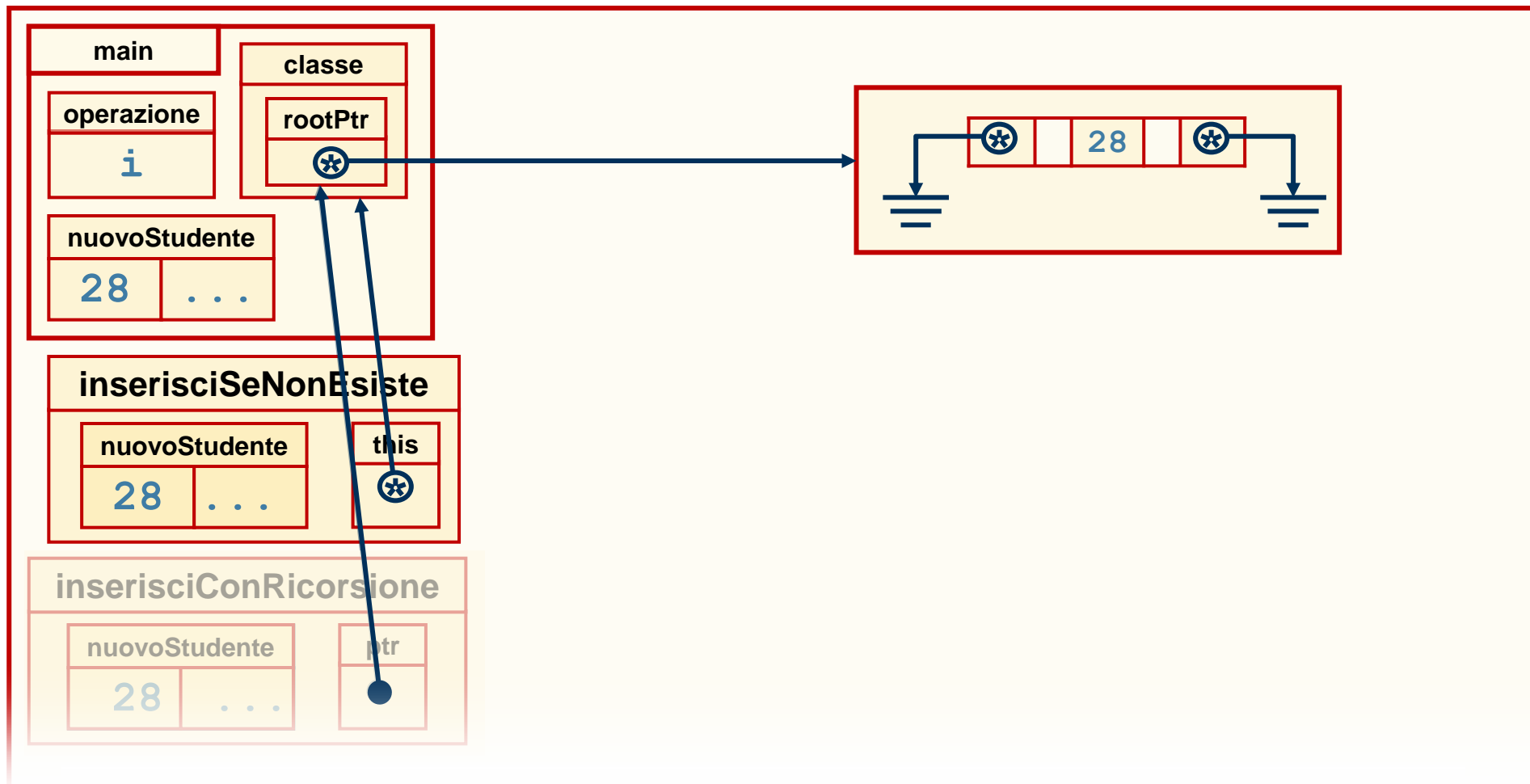
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```



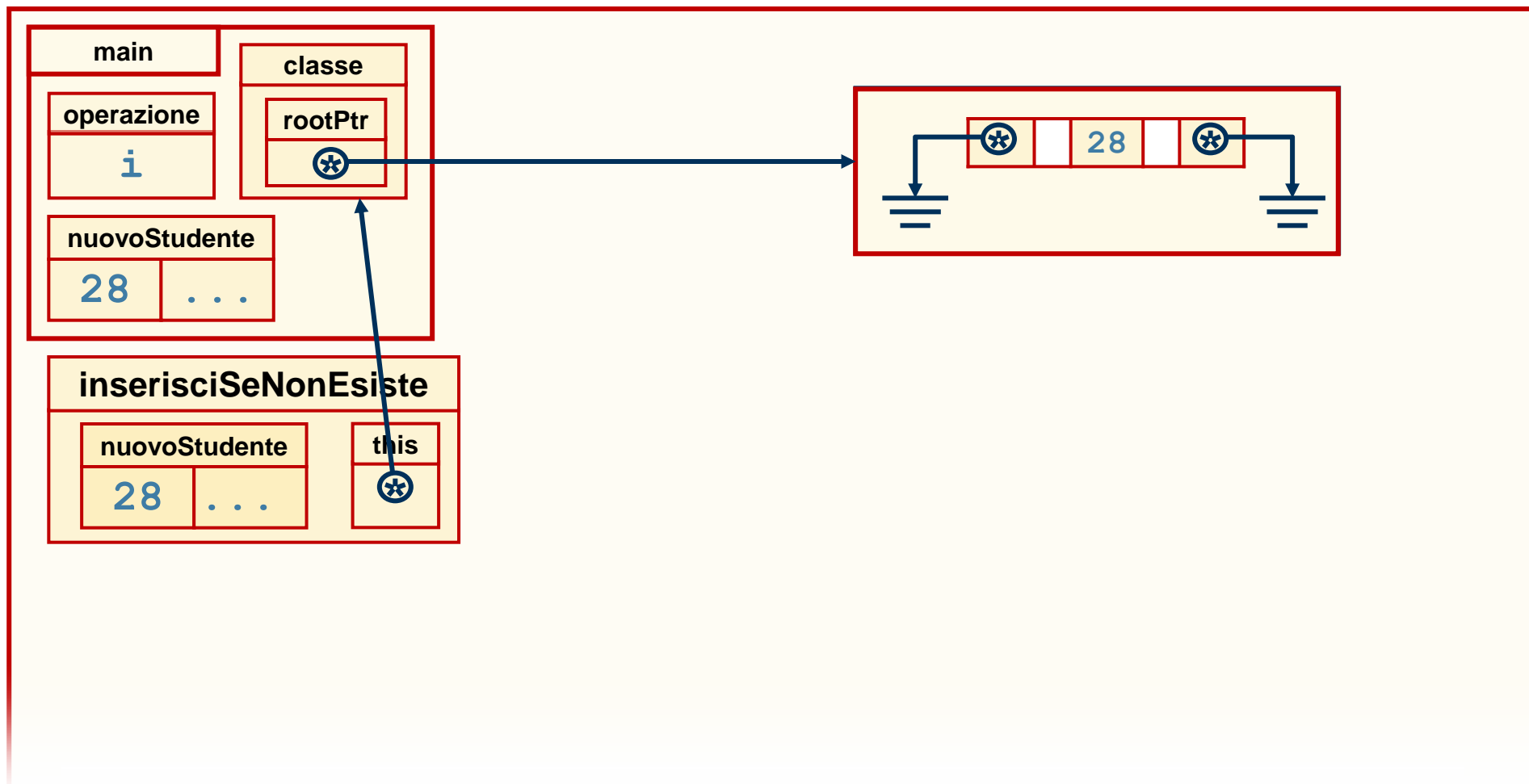
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente);
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```



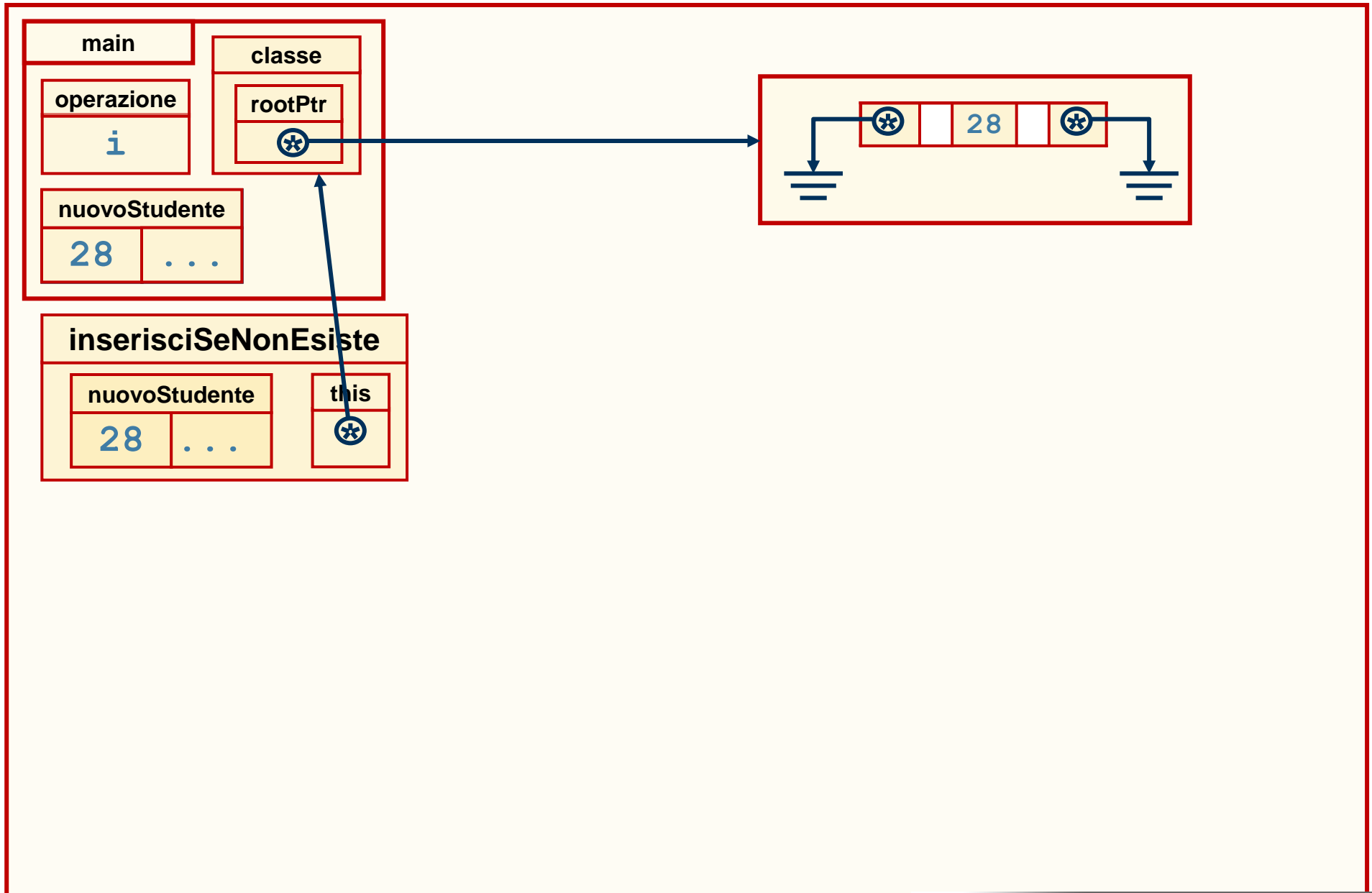
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente) ;
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```

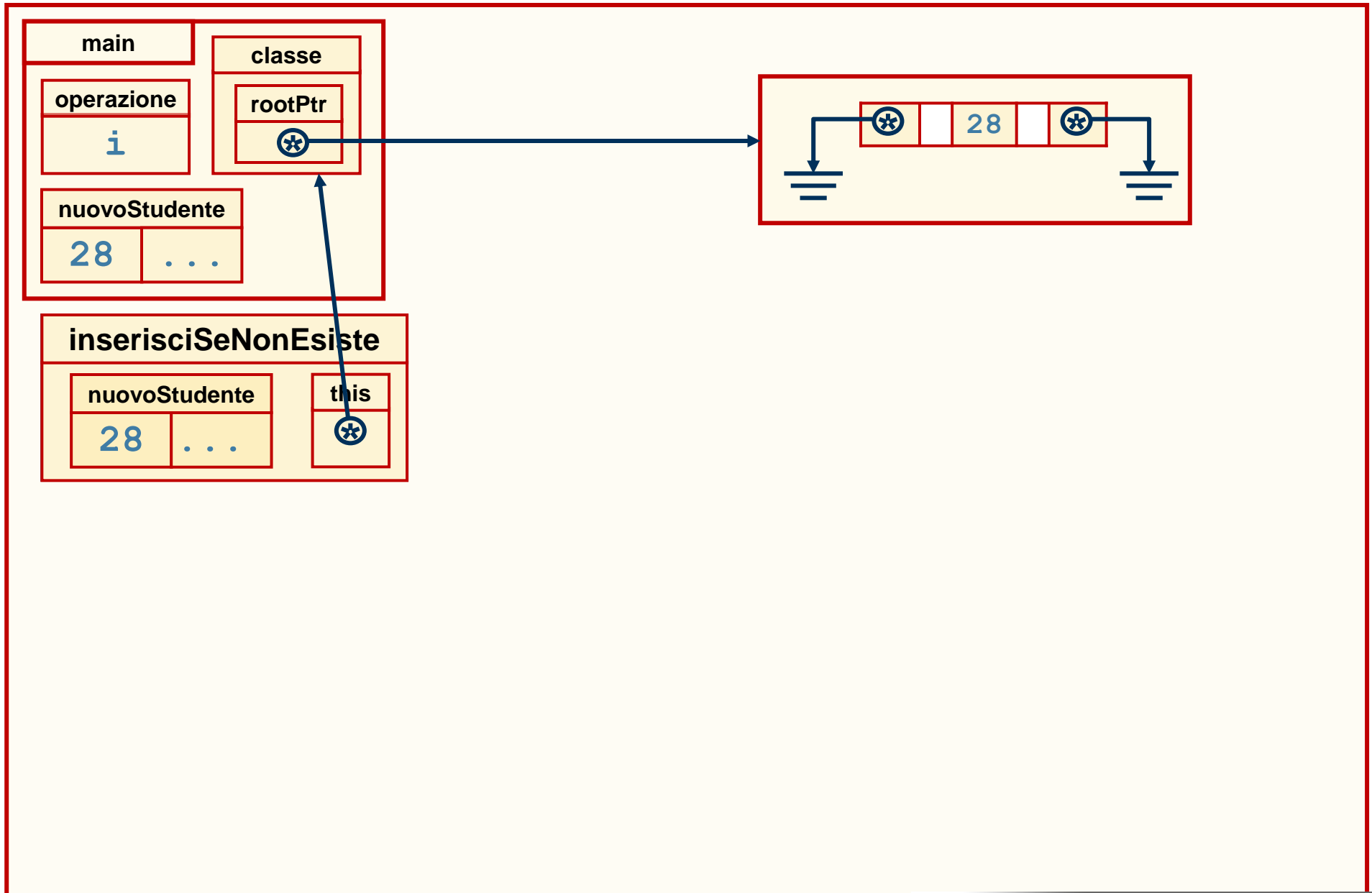


```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
  { ptr = new TreeNode(nuovoStudente);
  }
  else if (nuovoStudente.matricola < ptr->datiStud
```

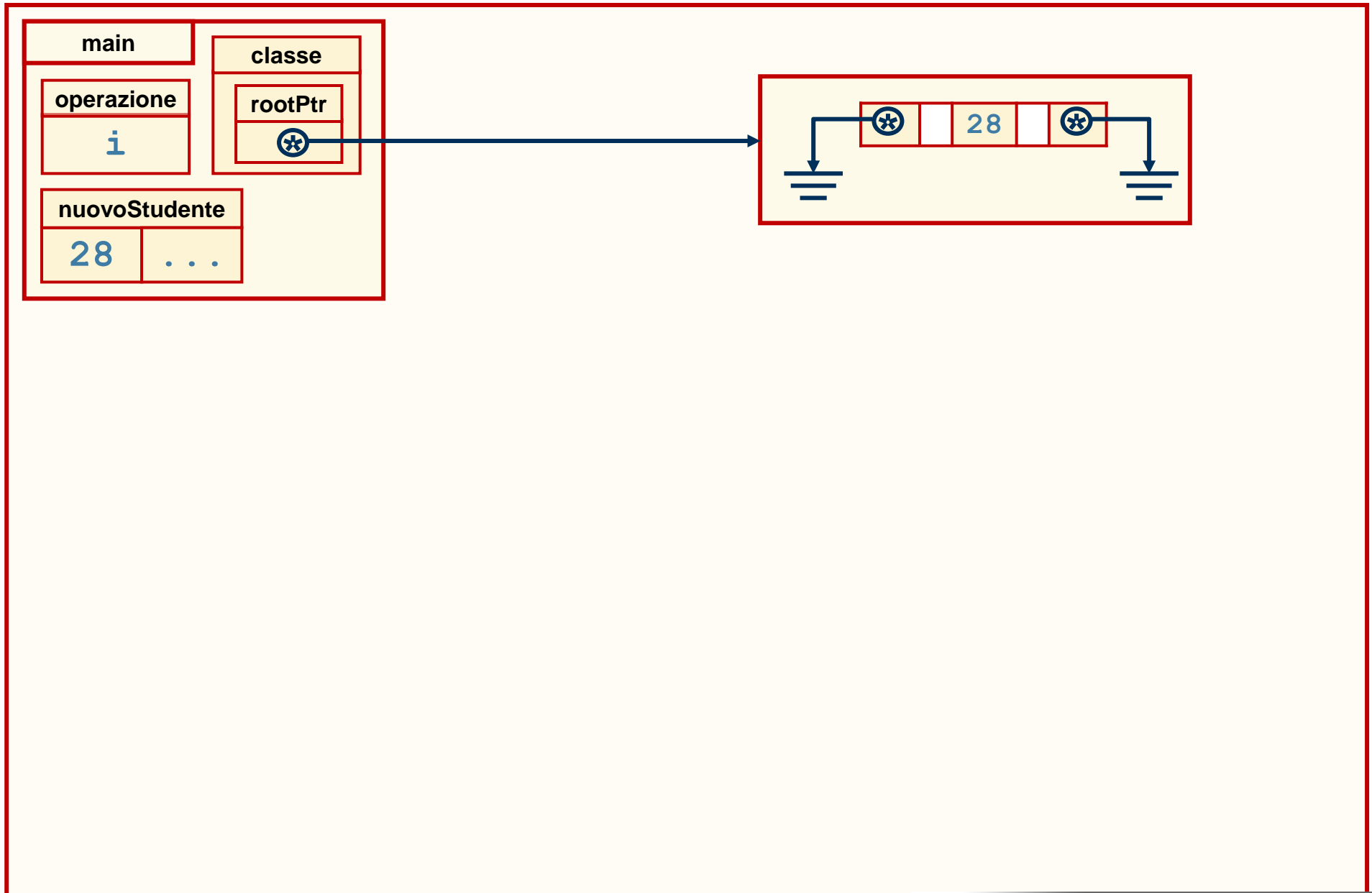


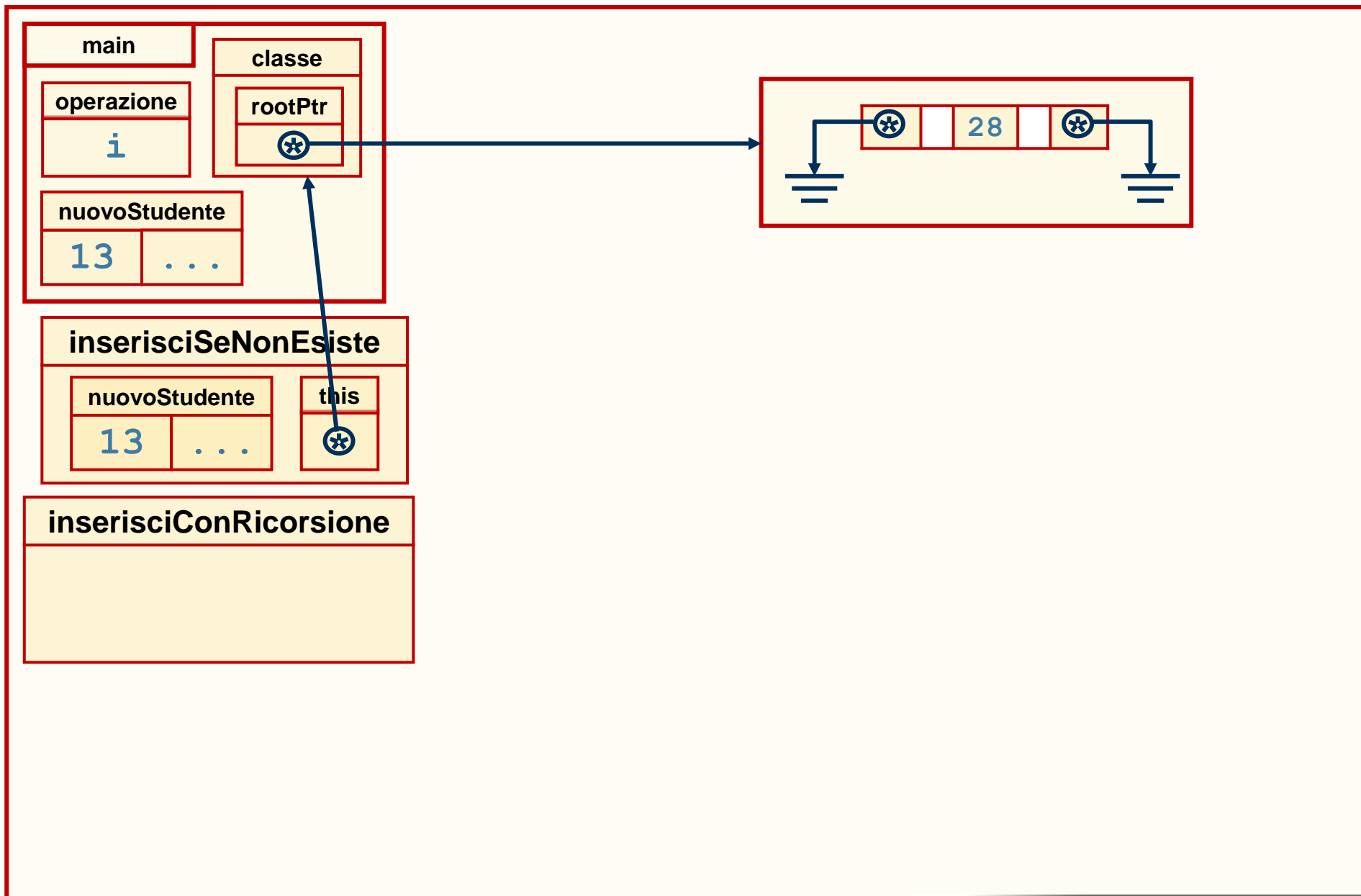
```
void Tree :: inserisciConRicorsione(TreeNode *& ptr,
{ if (ptr == 0)
    { ptr = new TreeNode(nuovoStudente) ;
    }
    else if (nuovoStudente.matricola < ptr->datiStud
```

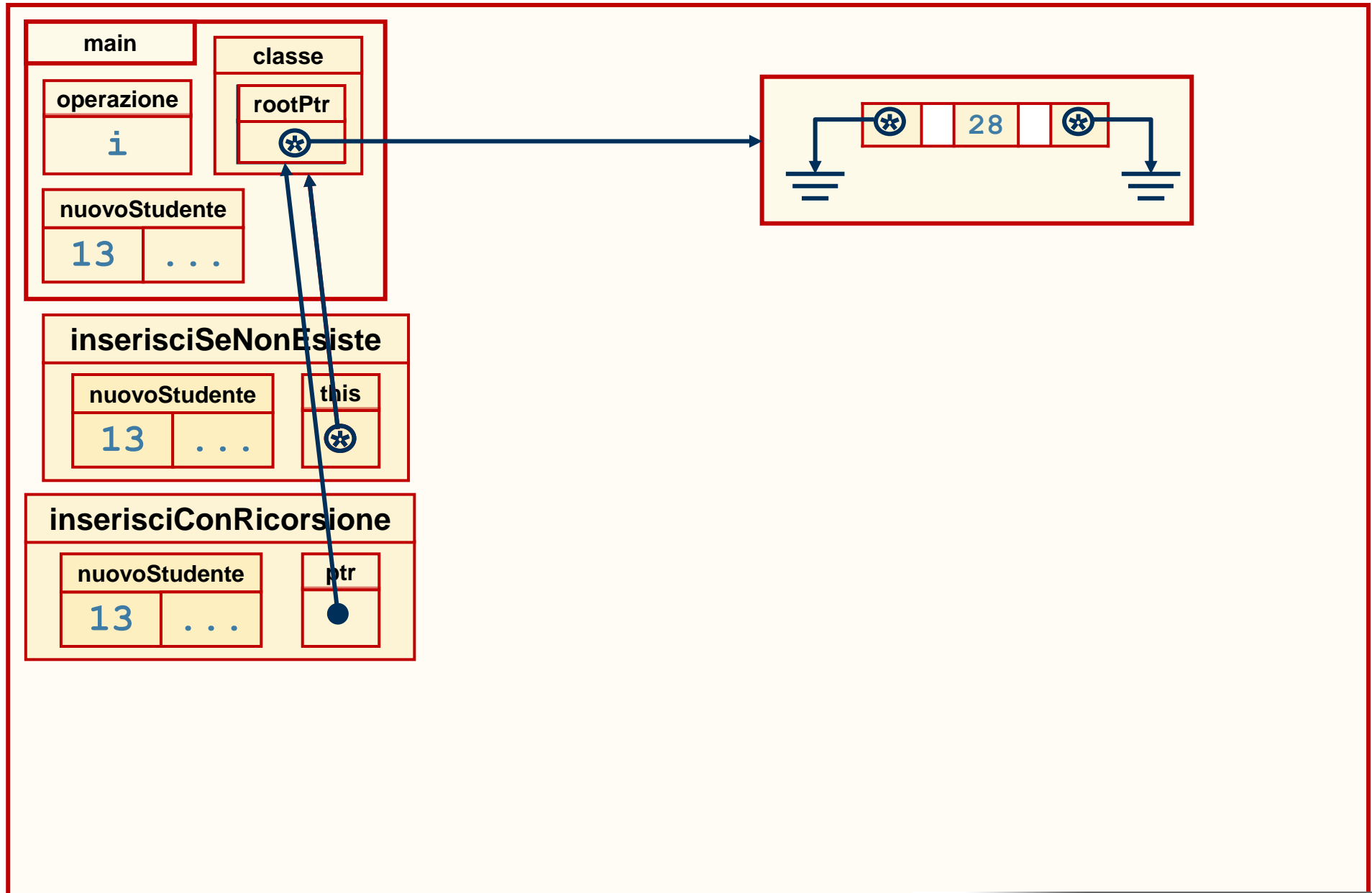


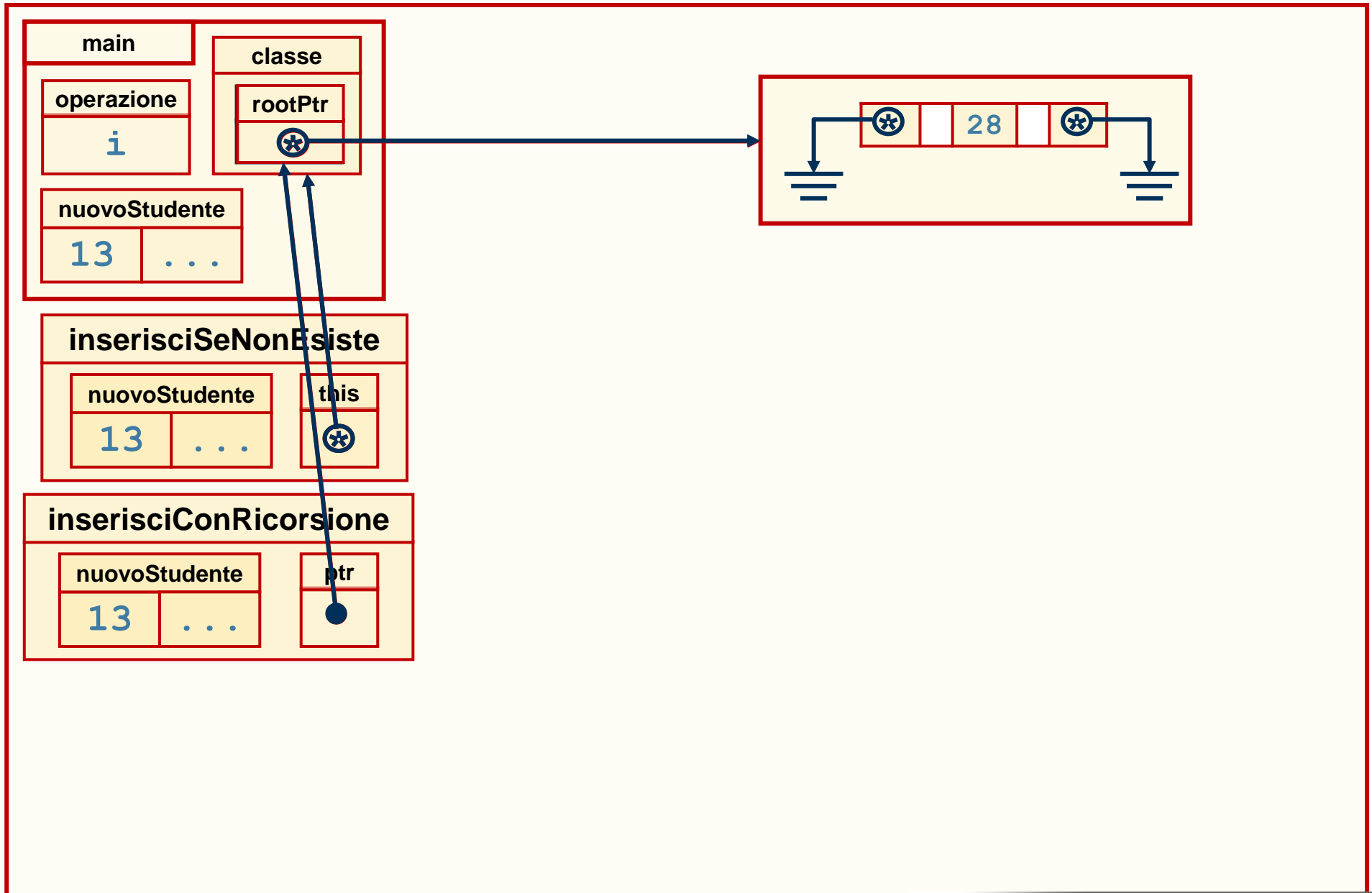


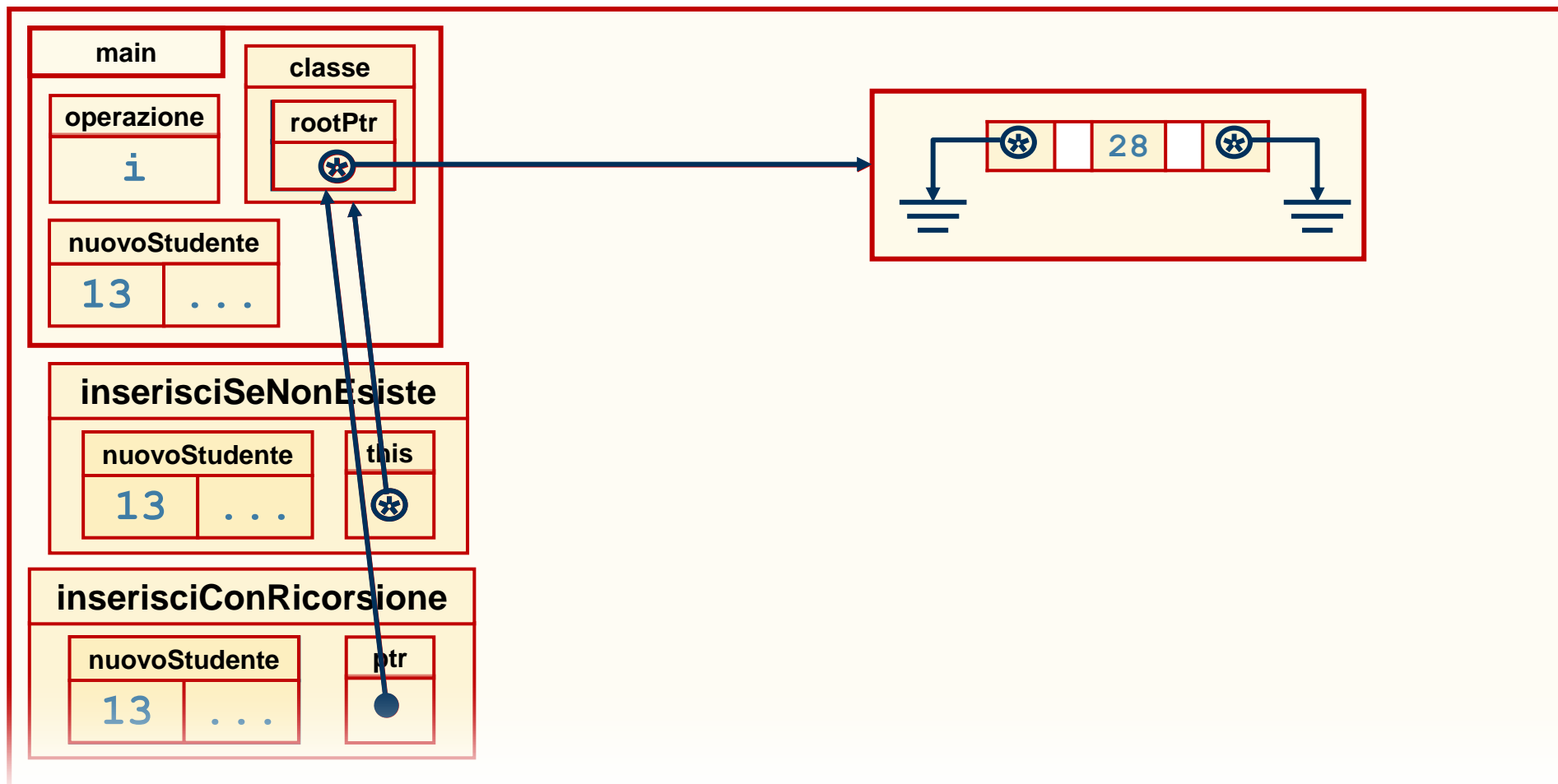




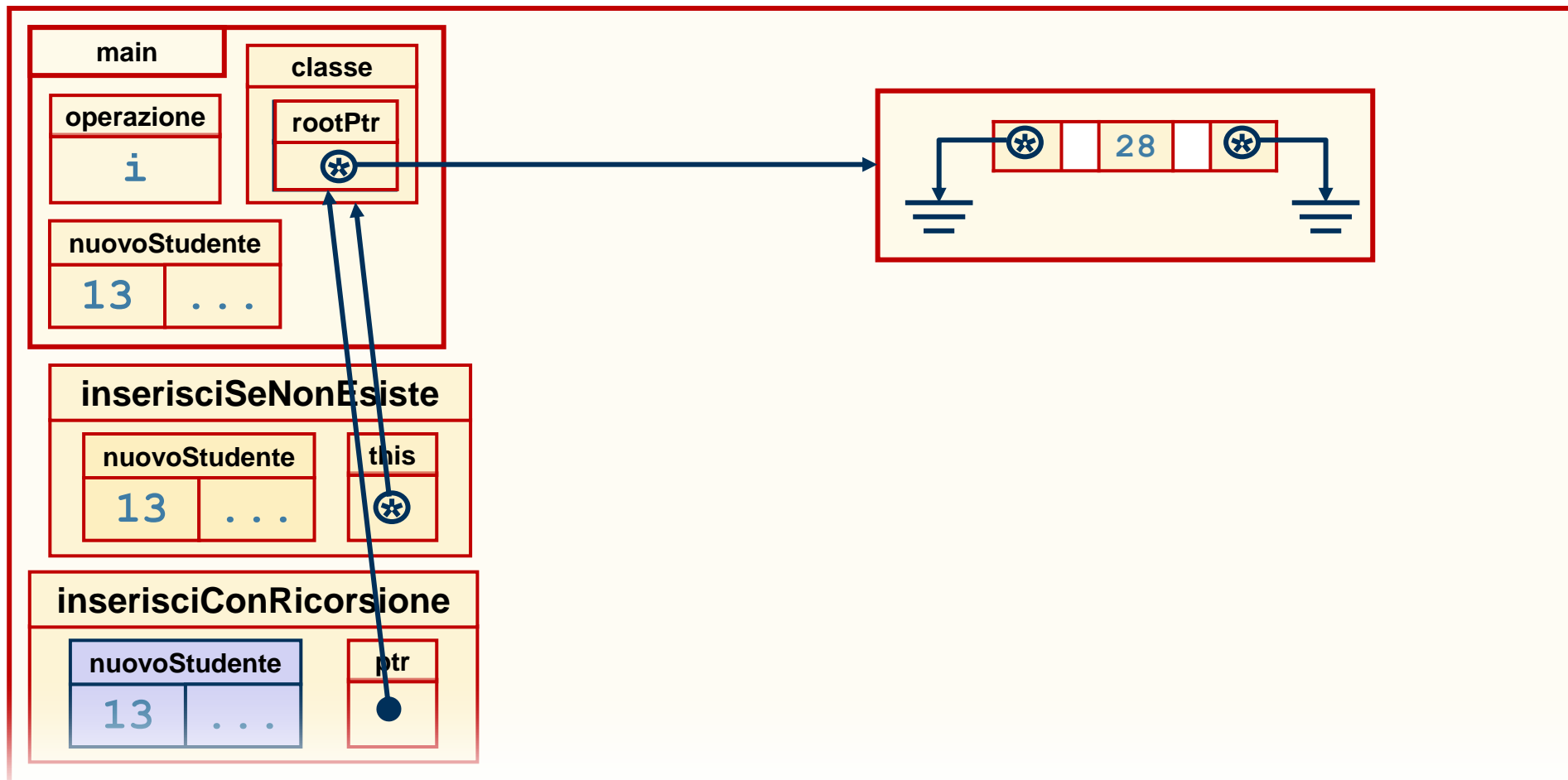








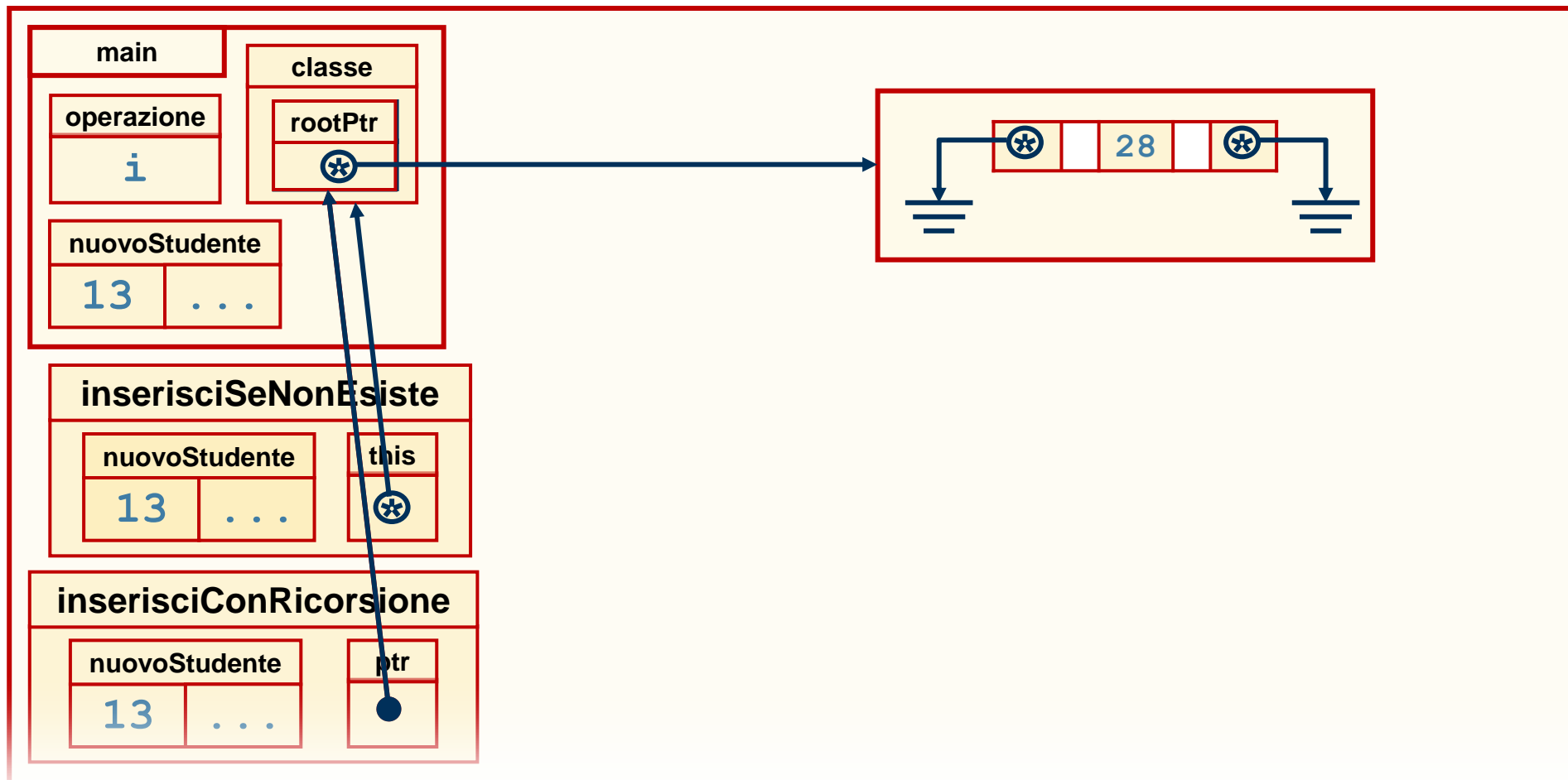
```
{ ptr = new TreeNode(nuovoStudente);  
}  
else if (nuovoStudente.matricola < ptr->datiStud.matricola)  
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);  
else if (nuovoStudente.matricola > ptr->datiStud.matricola)
```



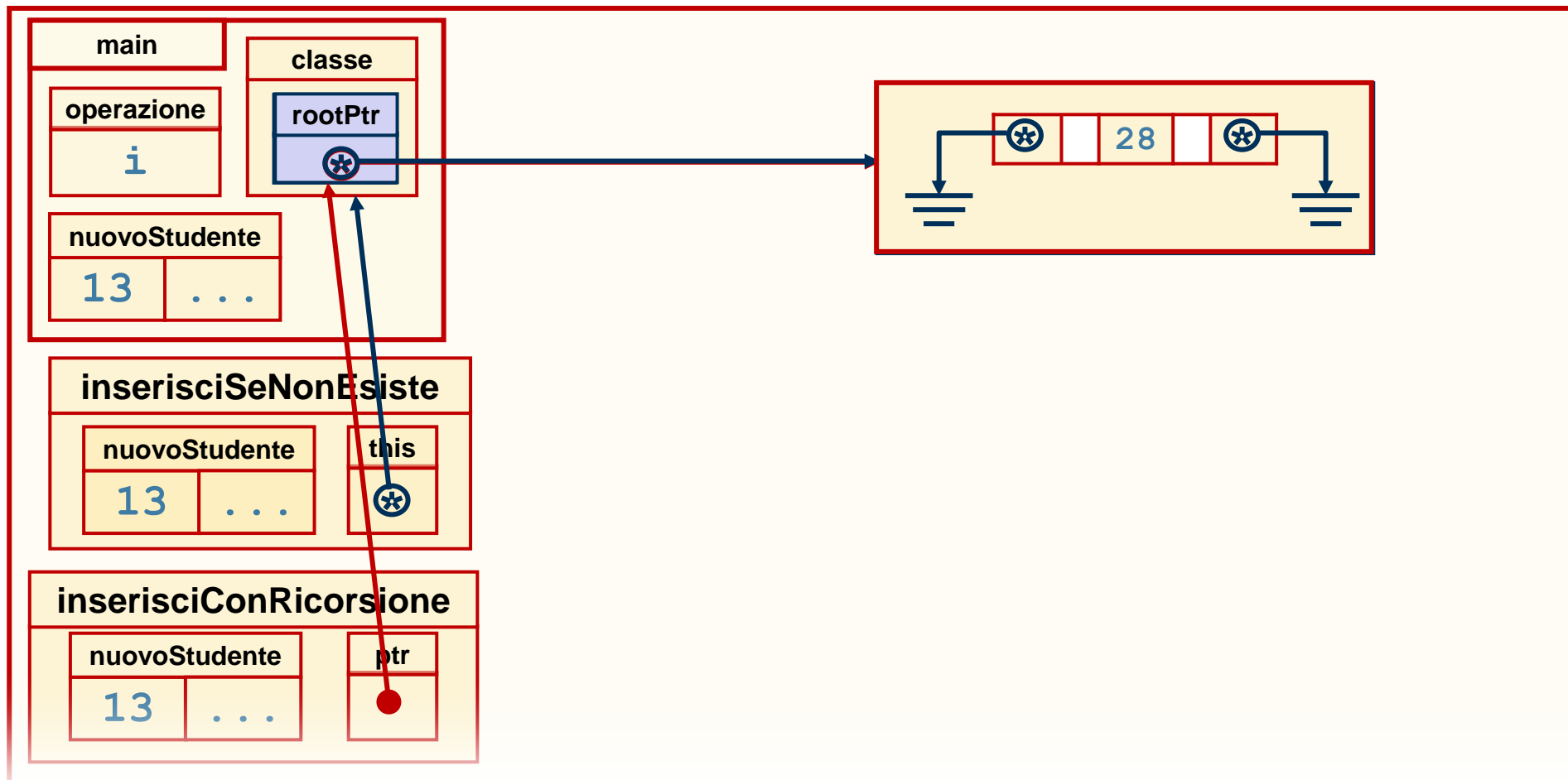
```

{ ptr = new TreeNode(nuovoStudente);
}
else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);
else if (nuovoStudente.matricola > ptr->datiStud.matricola)

```

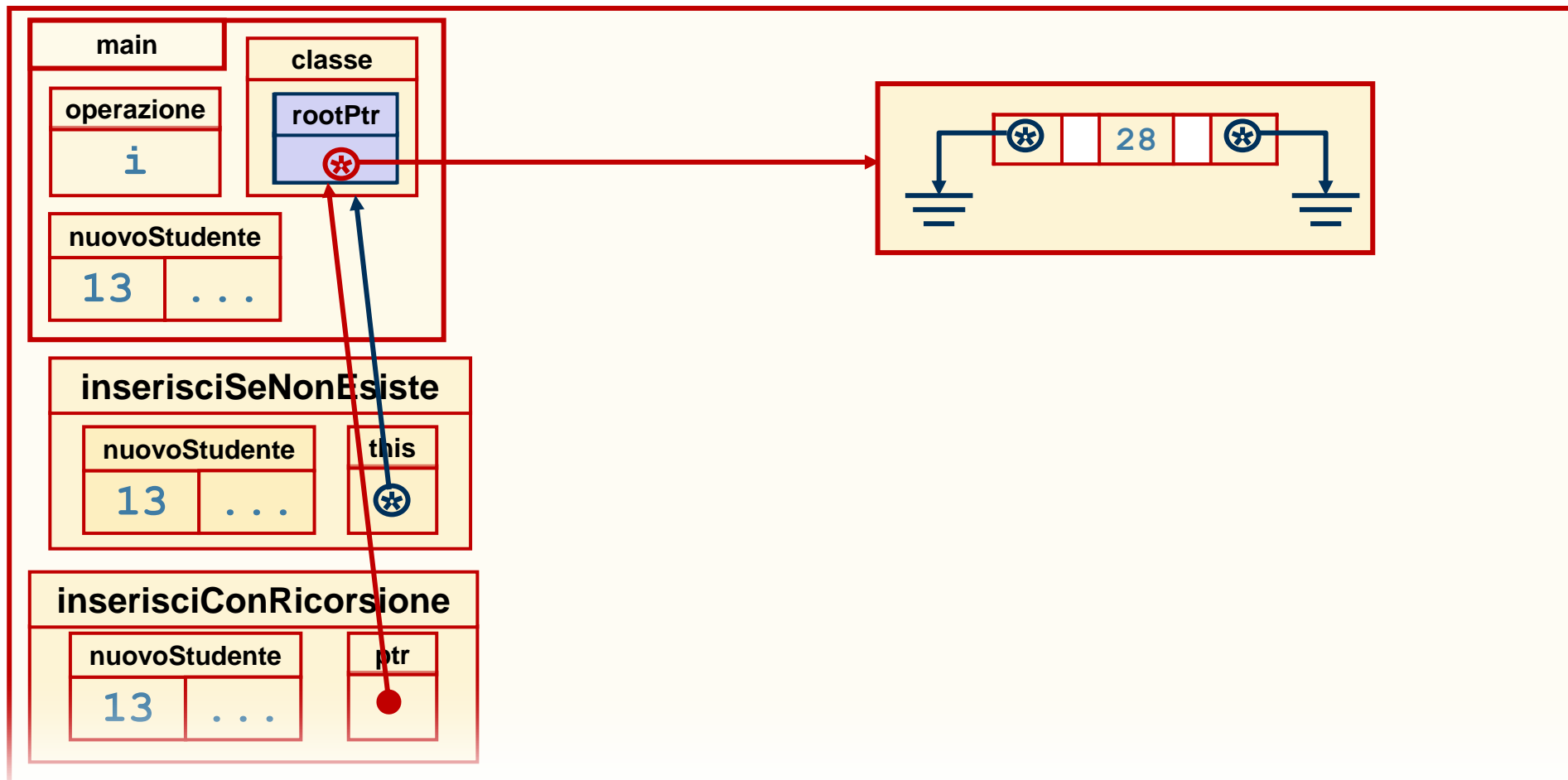


```
{ ptr = new TreeNode(nuovoStudente);  
}  
else if (nuovoStudente.matricola < ptr->datiStud.matricola)  
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);  
else if (nuovoStudente.matricola > ptr->datiStud.matricola)
```



```
{ ptr = new TreeNode(nuovoStudiante);  
}  
else if (nuovoStudiante.matricola < ptr->datiStud.matricola)  
    inserisciConRicorsione (ptr->leftPtr, nuovoStudiante);  
else if (nuovoStudiante.matricola > ptr->datiStud.matricola)
```

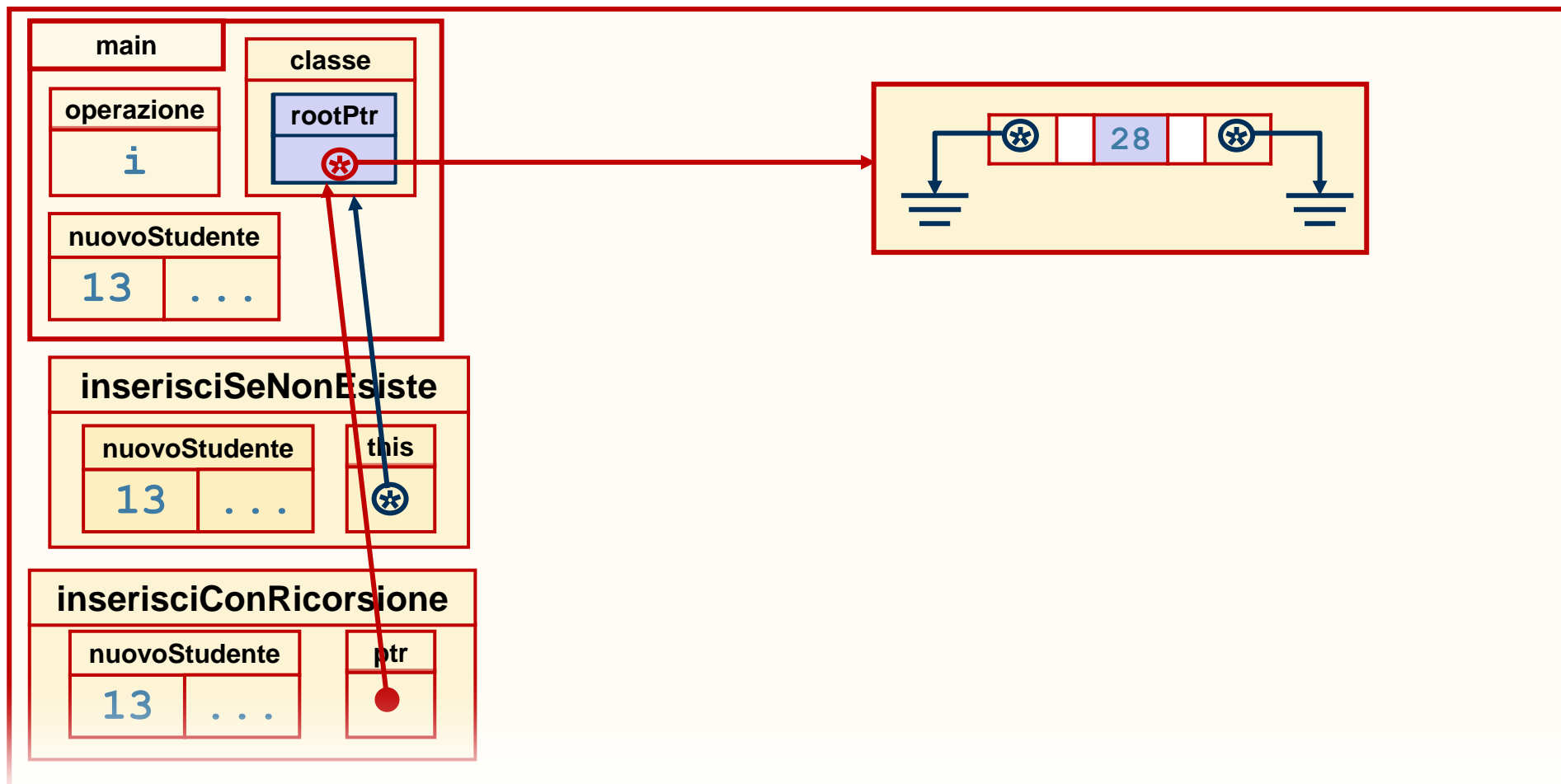




```

{ ptr = new TreeNode(nuovoStudente);
}
else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);
else if (nuovoStudente.matricola > ptr->datiStud.matricola)

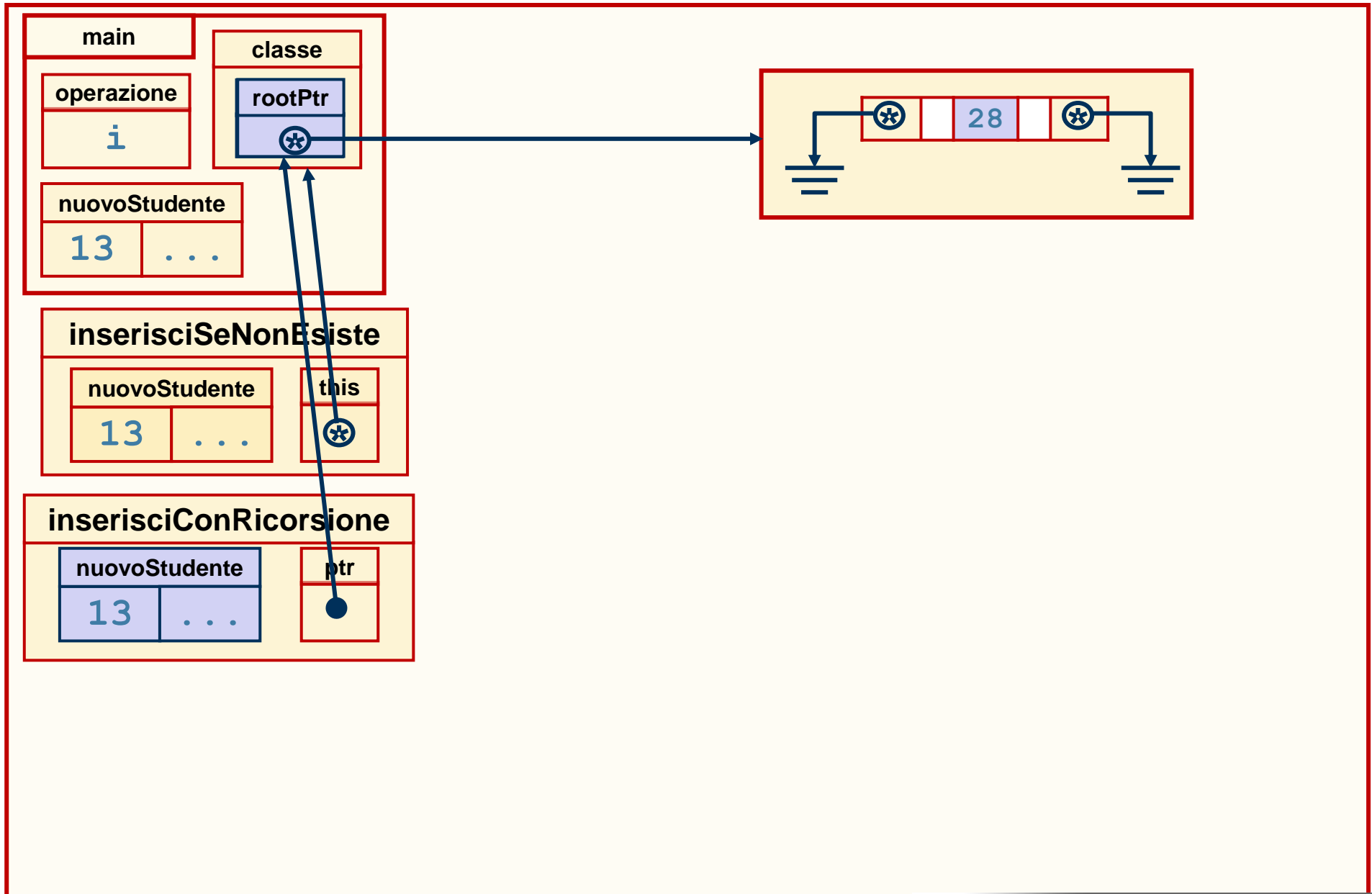
```



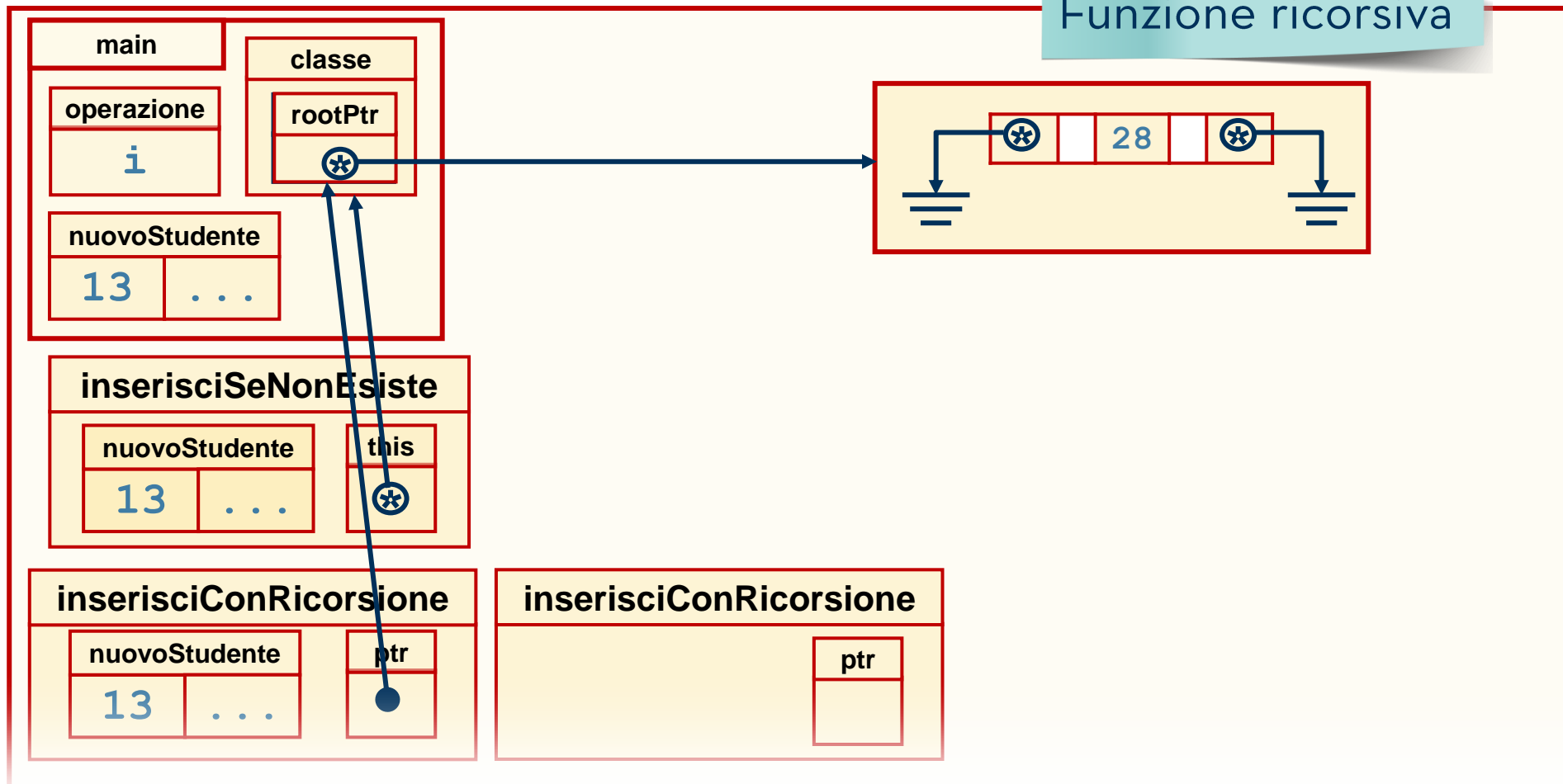
```

{ ptr = new TreeNode(nuovoStudente);
}
else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);
else if (nuovoStudente.matricola > ptr->datiStud.matricola)

```



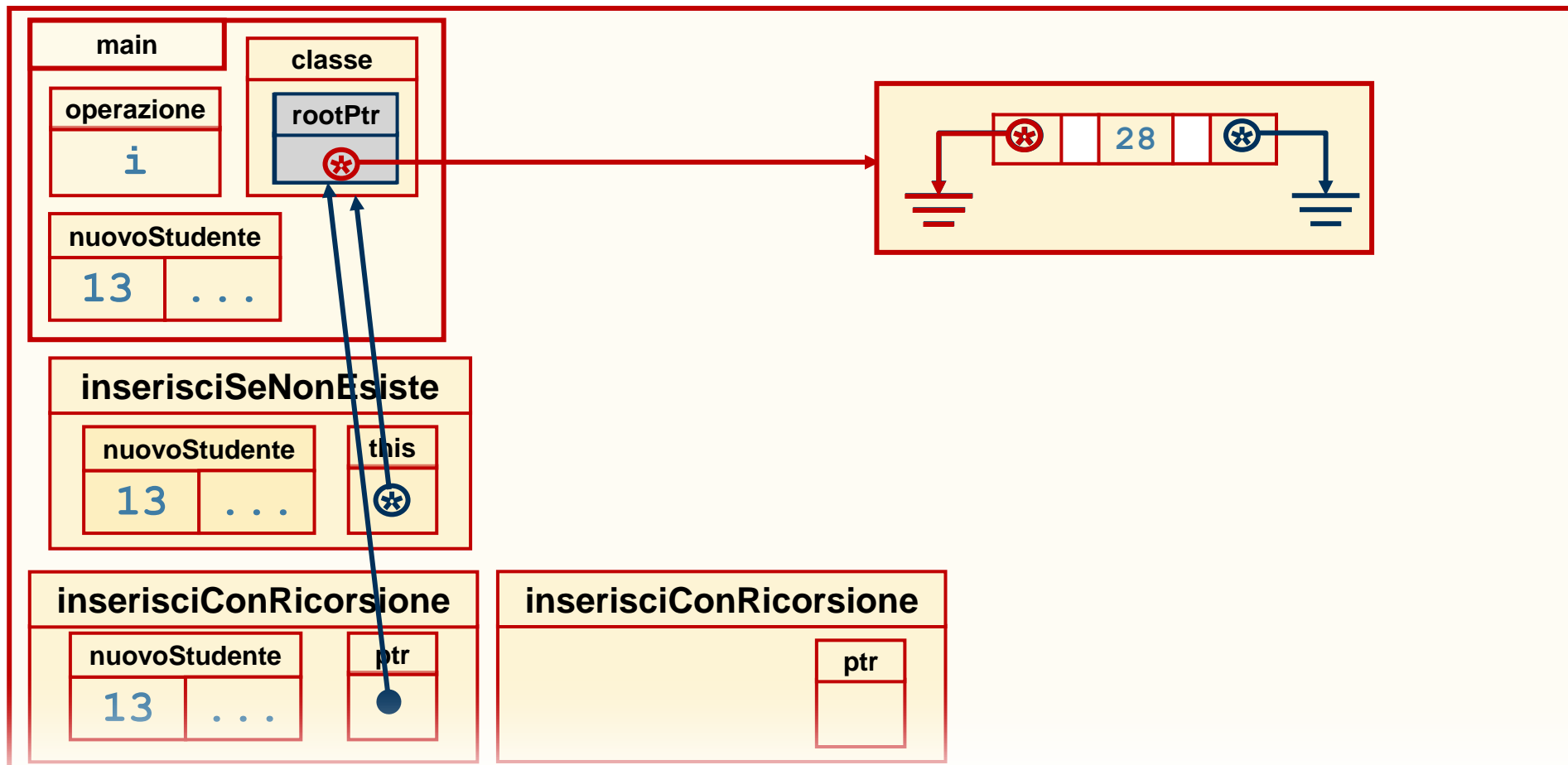
Funzione ricorsiva



```

}
else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);
else if (nuovoStudente.matricola > ptr->datiStud.matricola)

```



```

}
else if (nuovoStudente.matricola < ptr->datiStud.matricola)
    inserisciConRicorsione (ptr->leftPtr, nuovoStudente);
else if (nuovoStudente.matricola > ptr->datiStud.matricola)

```

