```
/*
 *  La classe Capitale représente les Villes capitales de leur pays
 *
 * @author Vincent Granet (vg@unice.fr)
 * @version 1.0
 *
 *    Creation @date: 19-Jan-2017 15:34
 *  Last file update:  8-Feb-2017 17:26
 */

#ifndef CAPITALE_HPP
#define CAPITALE_HPP

#include <string>

#include "Ville.hpp"
#include "Pays.hpp"

class Capitale : public Ville {
private:
  typedef Ville super;
public:
  // constructeurs
  Capitale(const std::string &v, const int n, Pays *p) : Ville(v,n,p) {
    Ville::pays->setCapitale(this);
  }
  //Pays par défaut la France
  Capitale(const std::string &v, const int n) : Ville(v,n) {
    if (v.compare("Paris")!=0) super::pays=nullptr;
  }
  /*
   * Rôle : renvoie une représentation de l'objet courant
   *        sous forme d'une chaîne de caractères
   */
  virtual std::string toString() const {
    return "*" + super::toString();
  }
};

#endif
```



```
/*
 *  Programme de test  pour les classes Pays, Ville et Capitale
 *  Elec4 : DS du 18 janvier 2017
 *
 * @author Vincent Granet (vg@unice.fr)
 * @version 1.0
 *
 *    Creation @date: 19-Jan-2017 15:34
 *  Last file update:  8-Feb-2017 17:15
 */

#include <iostream>
#include <vector>
#include <cstdlib>

#include "Ville.hpp"
#include "Pays.hpp"
#include "Capitale.hpp"

int main() {

  Pays *p = new Pays("Italie");

  std::vector<Ville*> lesVilles = {
    new Ville("Nice", 300000),
    new Capitale("Paris", 2500000),
    new Ville("Hambourg", 1700000),
    new Ville("Xian", 8000000, new Pays("Chine")),
    new Capitale("Rome", 2500000, p),
    new Capitale("Bruxelles", 200000),
  };

  lesVilles[2]->setPays(new Pays("Allemagne"));
  lesVilles[lesVilles.size()-1]->setPays(new Pays("Belgique"));

  for (auto v : lesVilles) std::cout << *v << std::endl;
  //
  std::cout << *p << std::endl;

  try {
    Pays *brexit = new Pays("Angleterre");
    std::cout << *brexit << std::endl;

  }
  catch(PaysException &e) {
    std::cerr << "Exception :" << e.what() << std::endl;
  }
  catch(VilleException &e) {
    std::cerr << "Exception :" << e.what() << std::endl;
  }

  return EXIT_SUCCESS;
}
```

```cpp
#include <iostream>
#include <string>

#include "Pays.hpp"
#include "Capitale.hpp"

Pays* const Pays::FRANCE = new Pays("France");

/*
 * Rôle : détruit l'objet courant avec sa Capitale
 */
Pays::~Pays() { delete this->capitale; }

// les accesseurs

/*
 * Rôle : renvoie le nom du pays courant
 */
std::string Pays::getNom() const {
  return this->nom;
}

/*
 * Rôle : renvoie, si elle existe, la capitale du pays courant
 *        sinon émet une exception
 */
Capitale *Pays::getCapitale() const throw (PaysException) {
  if (this->capitale == nullptr) throw PaysException();
  // ok le Pays courant possède une Capitale
  return this->capitale;
}

// les mutateurs

/*
 * Rôle : affecte le nom du pays courant
 */
void Pays::setNom(std::string n) {
  this->nom = n;
}

/*
 * Rôle : affecte la capitale du pays courant
 */
void Pays::setCapitale(Capitale *c) {
    this->capitale = c;
}

/*
 * Rôle : renvoie une représentation de l'objet courant
 *        sous forme d'une chaîne de caractères
 */
std::string Pays::toString() const {
  return "[" + this->nom + "-" + this->getCapitale()->getNom() + "]";
}

/*
 *  Rôle : renvoie *this = p
 */
Pays &Pays::operator=(const Pays &p) {
    this->nom = p.nom;
    this->capitale = p.capitale;
    return *this;
  }

/*
```



```cpp
 * Rôle : écrit sur le flot f le pays p
 */
std::ostream &operator<<(std::ostream & f, const Pays &p) {
  return f << p.toString();
}
```

```
/*
 *  La classe Pays représente un pays avec sa capitale
 *
 * @author Vincent Granet (vg@unice.fr)
 * @version 1.0
 *
 *    Creation @date: 19-Jan-2017 15:34
 *  Last file update:  2-Feb-2017 17:44
 */

#ifndef PAYS_HPP
#define PAYS_HPP

#include <iostream>
#include <string>
#include <exception>

#include "Pays.hpp"

class PaysException : public std::exception {
public:
  virtual const char* what() const throw() {
    return "Pays sans capitale !";
  }
};

class Capitale;

class Pays {
protected:
  std::string nom;
  Capitale *capitale;
public:
  //
  static Pays* const FRANCE;
  // constructeur
  Pays(const std::string &n, Capitale *c=nullptr) : nom(n),  capitale(c) {}
  // constructeur de copie
  Pays(const Pays &p) : nom(p.nom),  capitale(p.capitale) {}
  // destructeur (nécessaire, il détruit la Capitale)
  ~Pays();
  // les accesseurs
  std::string getNom() const;
  Capitale *getCapitale() const throw (PaysException);
  // les mutateurs
  void setNom(std::string n);
  void setCapitale(Capitale *c);
  // opérateur d'affectation
  Pays &operator=(const Pays &p);
  // pour l'écriture
  std::string toString() const;
  friend std::ostream &operator<<(std::ostream & f, const Pays &p);
};

#endif
```



```
#include <iostream>
#include <string>
#include <cassert>

#include "Ville.hpp"

// accesseurs

/*
 * Rôle : renvoie le nom de la ville courante
 */
std::string Ville::getNom() const {
  return this->nom;
}

/*
 * Rôle : renvoie le nombre d'habitants
 *        de la ville courante
 */
int Ville::getNbHabitants() const {
  return this->nbHabitants;
}

/*
 * Rôle : renvoie le pays auquel
 *        appartient la ville courante
 * Note : méthode virtuelle car utilisée dans toString
 */
Pays *Ville::getPays() const throw (VilleException) {
  if (this->pays==nullptr)  throw VilleException();
  // ok un Pays est associé à la ville courante
  return this->pays;
}

// mutateurs

/*
 * Rôle : affecte le nom de la ville courante
 */
void Ville::setNom(const std::string n) {
  this->nom = n;
}

/*
 * Rôle : affecte le nombre d'habitants
 *        de la ville courante
 */
void Ville::setNbHabitants(const int n) {
  assert(n>=0);
  this->nbHabitants = n;
}

/*
 * Rôle : affecte le pays auquel
 *        appartient la ville courante
 */
void Ville::setPays(Pays *p) {
  this->pays = p;
}

/*
 * Rôle : renvoie une représentation de l'objet courant
 *        sous forme d'une chaîne de caractères
 */
std::string Ville::toString () const {
  return this->nom + "(" + this->getPays()->getNom() + ")-"
```

```
    + std::to_string(this->nbHabitants) + "h";
}

/*
 *  Rôle : renvoie *this = p
 */
Ville &Ville::operator=(const Ville &v) {
  this->nom = v.nom;
  this->nbHabitants = v.nbHabitants;
  this->pays = v.pays;
  return *this;
}

/*
 * Rôle : écrit la Ville v sur le flot f
 */
std::ostream &operator<<(std::ostream &f, const Ville &v) {
  return f << v.toString();
}
```



```
/*
 *  La classe Ville représente les villes d'un pays. Par défaut, une
 *  ville est créée avec la France comme pays.
 *
 * @author Vincent Granet (vg@unice.fr)
 * @version 1.0
 *
 *    Creation @date: 19-Jan-2017 15:34
 *  Last file update: 19-Dec-2017 14:29
 */

#ifndef VILLE_HPP
#define VILLE_HPP

#include <iostream>
#include <string>
#include <exception>

#include "Pays.hpp"

class VilleException : public std::exception {
public:
  virtual const char* what() const throw() {
    return "Ville sans pays !";
  }
};
//
class Ville {
protected:
  std::string nom;
  int nbHabitants; // doit être >=0 (mais non testé)
  Pays *pays;
public:
  // constructeurs
  Ville(const std::string &v, const int n, Pays *p=Pays::FRANCE) :
    nom(v), nbHabitants(n), pays(p) {}

  // constructeur de copie
  Ville(const Ville &v) : nom(v.nom), nbHabitants(v.nbHabitants), pays(v.pays)
  {}
  // accesseurs
  std::string getNom() const;
  int getNbHabitants() const;
  virtual Pays *getPays() const throw (VilleException);
  // mutateurs
  void setNom(const std::string n);
  void setNbHabitants(const int n);
  void setPays(Pays *p);
  // opérateur d'affectation
 Ville &operator=(const Ville &v);
  //
  virtual std::string toString () const;
  friend  std::ostream &operator<<(std::ostream & f, const Ville &v);
};

#endif
```