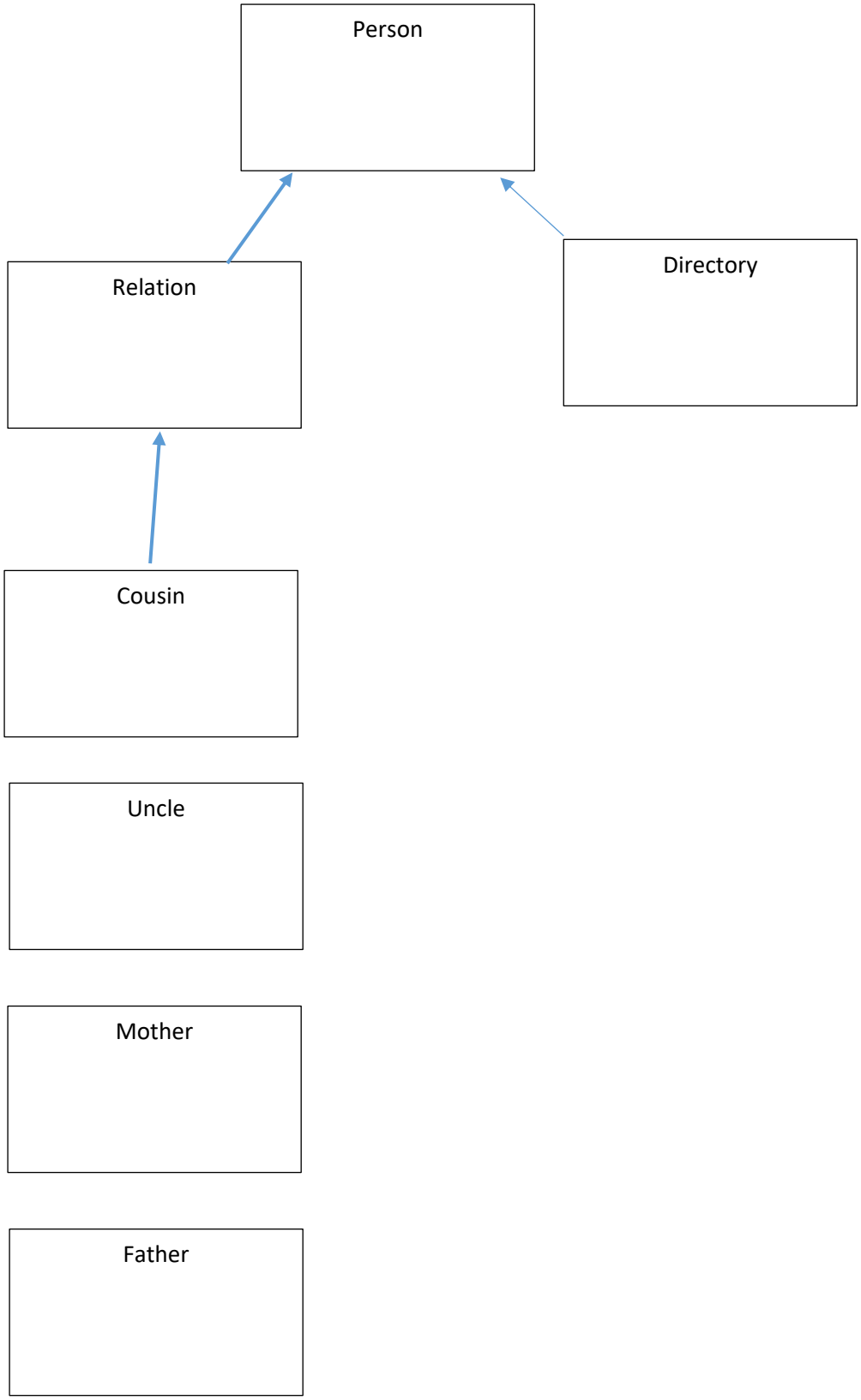
Code:

```
// Program used as the basis for DT021/3 computer labs.
// Each class holds some information and allows it to be changed
// and printed.

#include <stdio.h>
#include <string.h>


// Basic class to hold name, address and phone number.
class Person {
  private:
    char name[30];
    char addr[30];
    char tel[30];
  public:
    Person(char*,char*,char*);
    void setName(char*s) { strcpy(name,s); }
    void setAddr(char*s) { strcpy(addr,s); }
    void setNum(char*s) { strcpy(tel,s); }
    virtual void print();
};

Person :: Person(char*n,char*a,char*t) {
  strcpy(name,n);
  strcpy(addr,a);
  strcpy(tel,t);
}

void Person :: print() {
  printf("Name=%s, Addr=%s, Tel=%s\n",name,addr,tel);
}


// Subclass to add relationship and age.
class Relation : public Person {
  private:
    char relship[30];
    int age;
  public:
    Relation(char*,char*,char*,char*,int);
    void setRel(char* s) { strcpy(relship,s); }
    void setAge(int a) { age = a; }
    virtual void print();
};


Relation::Relation(char*n,char*a,char*t,char*r,int g) :
Person(n,a,t) {
  strcpy(relship,r);
  age = g;
}

void Relation::print() {
  Person::print();
  printf("Relationship=%s, Age=%d\n", relship, age);
}

//--------------------Uncle Realtionship subclass------------------------
// Illustrate subclassing one more level.
```

```
class Uncle : public Relation {
  private:
    void setRel(char*s) {}  // Override inheritance to block use.
  public:
    Uncle(char*n,char*a,char*t,int g) : Relation(n,a,t,"Uncle",g) {}
};
//-------------------Cousin Realtionship subclass----------------------
class Cousin : public Relation {
  private:
    void setRel(char*s) {}  // Override inheritance to block use.
  public:
    Cousin(char*n,char*a,char*t,int g) : Relation(n,a,t,"Cousin",g) {}
};

//-------------------Mother Realtionship subclass----------------------
class Mother : public Relation {
  private:
    void setRel(char*s) {}  // Override inheritance to block use.
  public:
    Mother(char*n,char*a,char*t,int g) : Relation(n,a,t,"Mother",g) {}
};

//-------------------Father Realtionship subclass----------------------
class Father : public Relation {
  private:
    void setRel(char*s) {}  // Override inheritance to block use.
  public:
    Father (char*n,char*a,char*t,int g) : Relation(n,a,t,"Father",g) {}
};



//============================Directory======================
// Make a class to look after a set of Person objects.
class Directory {
  private:
    Person * list[20];
  public:
    Directory();
    void addEntry(Person *);
    void print();
};

Directory::Directory() {
  int n;
  for(n=0; n<20; n+=1) {
    list[n] = 0;
  }
}

void Directory::addEntry(Person *e) {
  int n;
  for(n=0; n<20; n+=1) {
    if(list[n] == 0) {
      list[n] = e;
      return;
    }
  }
}

void Directory::print() {
  int n;
```

```
  for(n=0; n<20; n+=1) {
    if(list[n] != 0) {
      list[n]->print();
    }
  }
}
//=========================================================

void main() {
  printf ("Using Person objects:\n");
  Person p1("Tom","Toontown","2345");
  Person p2("Jerry","Mousehole","8765");
  p1.print();
  p2.print();
  p1.setName("Bugs Bunny"); // Use functions.
  p2.setAddr("Unknown");
  p1.print();
  p2.print();

  printf("\nUsing Relation objects:\n");
  Relation r1("Fred","Bedrock","12345","Uncle",52);
  Relation r2("Wilma","Bedrock","98765","Cousin",22);
  r1.print();
  r2.print();
  r1.setName("Barney");            // Use inherited function.
  r1.print();

  printf("\nUsing Uncle and Cousin objects:\n");
  Uncle u1("Itchy","1 Springfield","34567",30);
  Cousin c1("Scratchy","2 Springfield","76543",10);
  Father f1("Tallat","118 Kickham Road","76543",48);
  Mother m1("Nusrat","118 kickham Road","76543",46);

  u1.print();
  c1.print();
  f1.print();
  m1.print();

//  u1.setRel("Aunt");      // Illustrate anomaly.
  u1.print();

  printf("\nUsing a Directory object for Person objects:\n");
  Directory dir;
  dir.addEntry(&p1);
  dir.addEntry(&p2);
  dir.print();

  // Illustrate polymorphism
  printf("\nUsing a Directory object for subclasses of Person:\n");
  dir.addEntry(&r1); dir.addEntry(&r2); dir.addEntry(&u1);
  dir.addEntry(&c1); dir.addEntry(&f1) ; dir.addEntry(&m1); dir.print();           //
What does it print?
```

Relation UML diagram:



Verification Compiler:

Z:\College\2019-2020\Software design\Lab 3 - Relations\Realations\Debug\Realations.exe

```
Using Person objects:
Name=Tom, Addr=Toontown, Tel=2345
Name=Jerry, Addr=Mousehole, Tel=8765
Name=Bugs Bunny, Addr=Toontown, Tel=2345
Name=Jerry, Addr=Unknown, Tel=8765

Using Relation objects:
Name=Fred, Addr=Bedrock, Tel=12345
Relationship=Uncle, Age=52
Name=Wilma, Addr=Bedrock, Tel=98765
Relationship=Cousin, Age=22
Name=Barney, Addr=Bedrock, Tel=12345
Relationship=Uncle, Age=52

Using Uncle and Cousin objects:
Name=Itchy, Addr=1 Springfield, Tel=34567
Relationship=Uncle, Age=30
Name=Scratchy, Addr=2 Springfield, Tel=76543
Relationship=Cousin, Age=10
Name=Tallat, Addr=118 Kickham Road, Tel=76543
Relationship=Father, Age=48
Name=Nusrat, Addr=118 kickham Road, Tel=76543
Relationship=Mother, Age=46
Name=Itchy, Addr=1 Springfield, Tel=34567
Relationship=Uncle, Age=30

Using a Directory object for Person objects:
Name=Bugs Bunny, Addr=Toontown, Tel=2345
Name=Jerry, Addr=Unknown, Tel=8765

Using a Directory object for subclasses of Person:
Name=Bugs Bunny, Addr=Toontown, Tel=2345
Name=Jerry, Addr=Unknown, Tel=8765
Name=Barney, Addr=Bedrock, Tel=12345
Relationship=Uncle, Age=52
Name=Wilma, Addr=Bedrock, Tel=98765
Relationship=Cousin, Age=22
Name=Itchy, Addr=1 Springfield, Tel=34567
Relationship=Uncle, Age=30
Name=Scratchy, Addr=2 Springfield, Tel=76543
Relationship=Cousin, Age=10
Name=Tallat, Addr=118 Kickham Road, Tel=76543
Relationship=Father, Age=48
Name=Nusrat, Addr=118 kickham Road, Tel=76543
Relationship=Mother, Age=46
```