Опис Person ADT:

Data:

* Стрічка з латинськими літерами та, можливо, цифрами
* Масив типу Person
* Елемент типу Person
* Елемент типу Person
* Елемент типу Person
* Tuple з двома цілими числами
* Символ “m” або “f”

Operations:

* Person(name, children, mother, father, partner, living\_years, sex)
* \_\_str\_\_()
* \_\_eq\_\_()
* \_\_ne\_\_()
* get\_mother()
* get\_father()
* get\_children()
* get\_name()
* get\_sex()
* get\_birth\_year()
* get\_death\_year()

Опис FamilyTree ADT:

Data:

* об’єкт типу Person

Operations:

* FamilyTree(root)
* draw()
* add\_child(person, child)
* add\_mother(person, mother)
* add\_father(person, father)
* add\_partner(person, partner)
* check\_marriage(man, woman)
* root(person)
* num\_children(person)
* children(person)
* mother(person)
* father(person)
* partner(person)

class Person:  
 *"""  
 Class representing a person.  
 """* def \_\_init\_\_(self, name, living\_years, sex, children=None, mother=None, father=None, partner=None):  
 *"""  
 Initialization of person.* ***:param*** *name: string* ***:param*** *living\_years: tuple wih 2 integers* ***:param*** *sex: "f" or "m"* ***:param*** *children: list of Persons* ***:param*** *mother: Persons with "f" sex* ***:param*** *father: Persons with "m" sex* ***:param*** *partner: Person with opposite sex  
 """*  
 def \_\_str\_\_(self):  
 *"""  
 Returns string representation of Person.* ***:return****: str  
 """*

def \_\_eq\_\_(self, other):  
 *"""  
 Returns True if self equals other.* ***:param*** *other: Person* ***:return****: bool  
 """*

def \_\_ne\_\_(self, other):  
 *"""  
 Returns True if self not equals other.* ***:param*** *other: Person* ***:return****: bool  
 """*  
 def get\_name(self):  
 *"""  
 Returns name of Person.* ***:return****: str  
 """*  
 def get\_sex(self):  
 *"""  
 Returns sex of Person.* ***:return****: str  
 """*  
  
 def get\_birth\_year(self):  
 *"""  
 Returns Person's birth year.* ***:return****: int  
 """*  
 def get\_death\_year(self):  
 *"""  
 Returns Person's death year.* ***:return****: int  
 """*  
 def get\_mother(self):  
 *"""  
 Returns Person's mother.* ***:return****: Person or None  
 """*  
 def get\_father(self):  
 *"""  
 Returns Person's father.* ***:return****: Person or None  
 """*  
 def get\_partner(self):  
 *"""  
 Returns Person's partner.* ***:return****: Person or None  
 """*  
 def get\_children(self):  
 *"""  
 Returns list of Person's children.* ***:return****: list of Persons or None  
 ""*

class FamilyTree:  
 *"""  
 Class representing family tree.  
 """* def \_\_init\_\_(self, root):  
 *"""  
 Initialization of tree based on main person.* ***:param*** *root: Person  
 """*  
 def draw(self):  
 *"""  
 Drawing a tree.* ***:return****: None  
 """*  
 def add\_child(self, person, child):  
 *"""  
 Add a child to given person.* ***:param*** *person: Person* ***:param*** *child: Person* ***:return****: None  
 """*  
 def add\_mother(self, person, mother):  
 *"""  
 Add a mother to given person.* ***:param*** *person: Person* ***:param*** *mother: Person* ***:return****: None  
 """*  
 def add\_father(self, person, father):  
 *"""  
 Add a father to given person.* ***:param*** *person: Person* ***:param*** *father: Person* ***:return****: None  
 """*  
  
 def add\_partner(self, person):  
 *"""  
 Add a child to given person.* ***:param*** *person: Person* ***:return****: None  
 """*  
 def check\_marriage(self, man, woman):  
 *"""  
 Checks if marriage between man and woman is possible.* ***:param*** *man: Person* ***:param*** *woman: Person* ***:return****: bool  
 """*  
 def root(self, person):  
 *"""  
 Checks if given person is a root.* ***:param*** *person: Person* ***:return****: bool  
 """*  
 def num\_children(self, person):  
 *"""  
 Returns number of person's children.* ***:param*** *person: Person* ***:return****: int  
 """*  
  
 def children(self, person):  
 *"""  
 Returns person's children.* ***:param*** *person: Person* ***:return****: list of Persons  
 """*  
  
 def mother(self, person):  
 *"""  
 Returns person's mother.* ***:param*** *person: Person* ***:return****: Person  
 """*  
  
 def father(self, person):  
 *"""  
 Returns person's father.* ***:param*** *person: Person* ***:return****: Person  
 """*  
  
 def partner(self, person):  
 *"""  
 Returns person's partner.* ***:param*** *person: Person* ***:return****: Person  
 """*