Trường Đại học Khoa học Tự nhiên

Khoa Toán – Cơ – Tin học

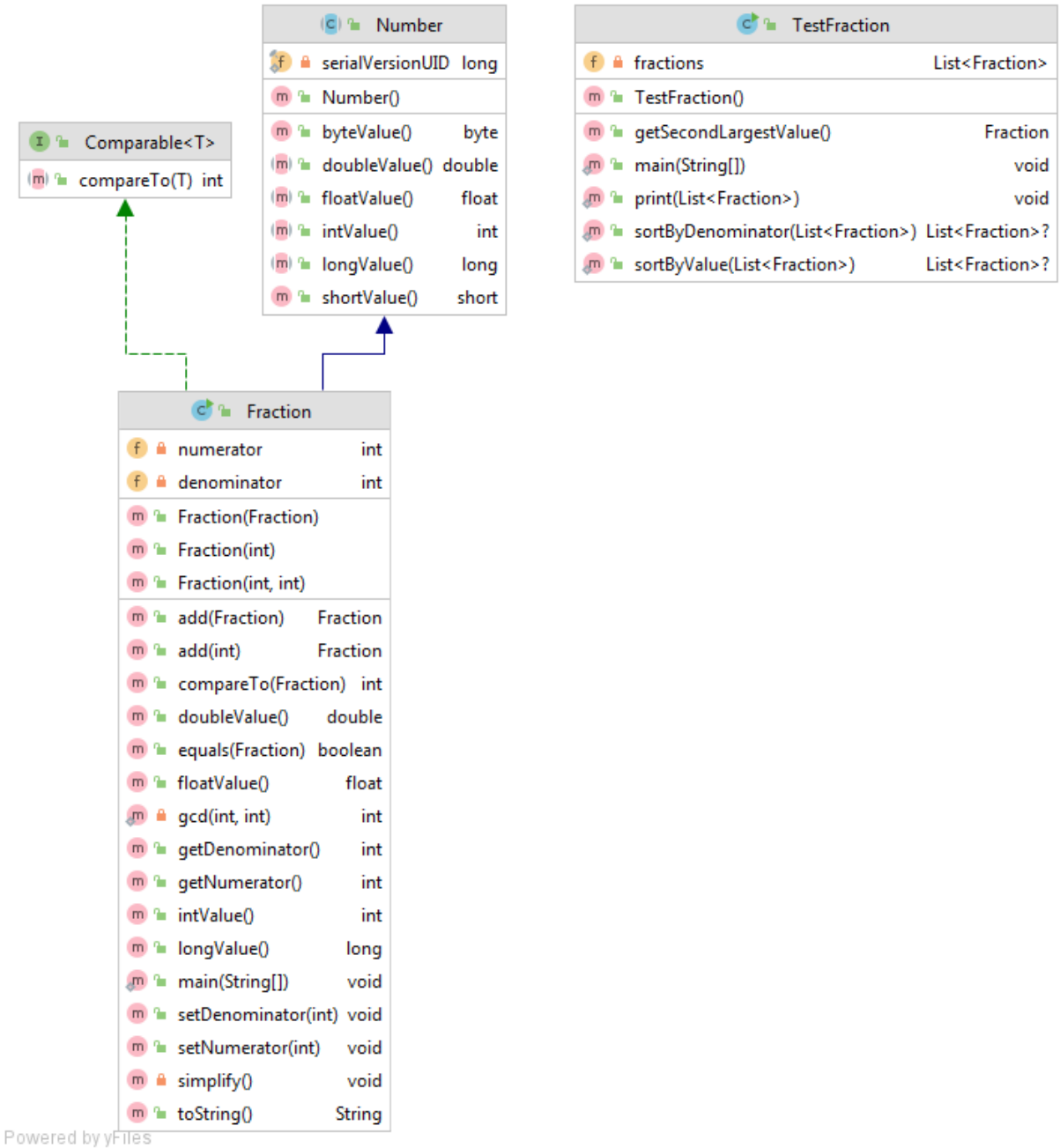
Thi giữa kỳ: Học kỳ II – 2021-2022

Học phần: Các Thành phần Phần mềm

Thời gian làm bài: 150 phút

Đề 1

Câu 1. Một chương trình Java được thiết kế như biểu đồ dưới đây.



Viết code cho các lớp của chương trình. Các lớp thuộc package “com.fraction”.

Chú thích:

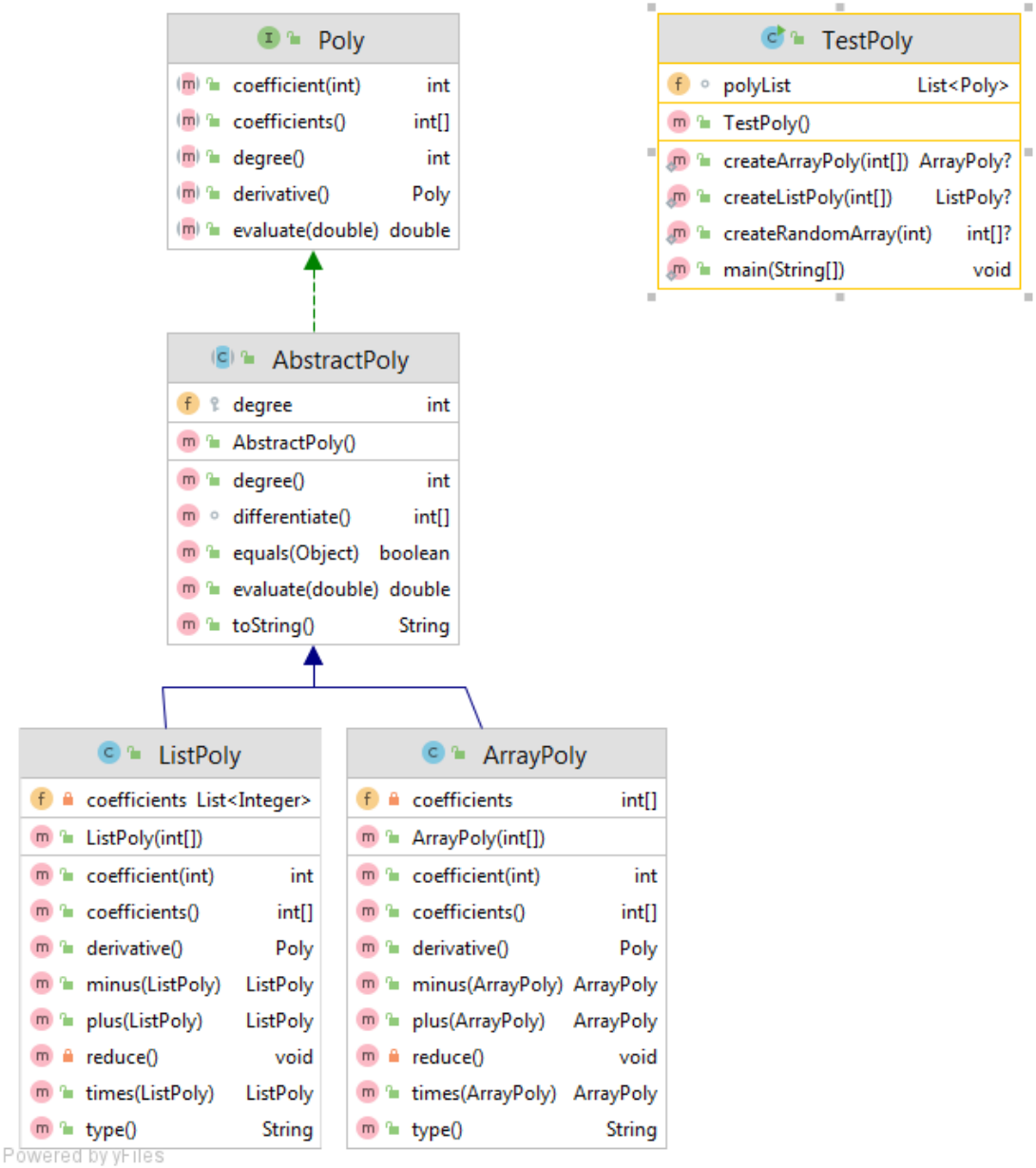
Lớp service **Fraction** mô tả một phân số:

- Lớp **Fraction** thừa kế từ lớp **java.lang.Number** và implements giao diện Comparable<Fraction>.
- Lớp **Fraction** có 2 thuộc tính là *numerator* và *denominator*.
- Hàm dựng *Fraction(int)* khởi tạo tử số theo tham số truyền vào, khởi tạo mẫu số bằng 1.
- Hàm *add(Fraction)* cộng hai *Fraction*, trả về *Fraction* mới là kết quả của cộng *Fraction* hiện tại với *Fraction* truyền vào.
- Hàm *add(int)* cộng hai *Fraction*, trả về *Fraction* mới là kết quả của cộng *Fraction* hiện tại với *Fraction* có tử số là tham số truyền vào, mẫu số là 1.
- Hàm *compareTo(Fraction)* overide hàm *compareTo* của *interface Comparable*, dùng để so sánh hai *Fraction*. *Fraction* A lớn hơn *Fraction* B nếu tử số của A nhân với mẫu số của B lớn hơn mẫu số của A nhân với tử số của B.
- Hàm *equals(Fraction)* kiểm tra hai *Fraction* có bằng nhau không. *Fraction* A bằng *Fraction* B nếu tử số của A nhân với mẫu số của B bằng mẫu số của A nhân với tử số của B.
- Hàm *gcd(int, int)* tính ước số chung lớn nhất của hai số. Nên dùng thuật toán Euclid để tính ước chung lớn nhất của hai số.
- Hàm *simplify()* rút gọn phân số đã cho về phân số tối giản.
- Hàm *toString()* trả về String mô tả phân số theo định dạng “Fraction[numerator/denominator]”.

Lớp test driver **TestFraction** thực hiện các yêu cầu sau:

- Tạo ngẫu nhiên 30 cặp số nguyên có giá trị nằm trong khoảng [1, 1000] để tạo ra 30 phân số *Fraction*, lưu các phân số này vào một List có tên là *fractions*.
- In ra các phân số trong list *fractions*; in ra các phân số theo thứ tự có giá trị tăng dần (theo sắp thứ tự của Comparable); in ra các phân số có mẫu số tăng dần; tìm phân số có giá trị lớn thứ hai trong list fractions. Mỗi thông tin được in trên một dòng.
- Việc sắp xếp các phân số có thể dùng các thuật toán sắp xếp đã học, ví dụ Selection Sort, Bubble Sort, …, hoặc dùng các giao diện Comparable và Comparator.

Bài 2. Một chương trình Java được thiết kế như biểu đồ dưới đây.



Viết code cho các lớp trong chương trình. Các lớp thuộc package “com.poly”.

Chú thích:

Các lớp service:

- **Poly** là một interface.
  - Phương thức *coefficient(int i)* trả về hệ số của x^i.
  - Phương thức *coefficients()* trả về một mảng là các hệ số của đa thức.
  - Phương thức *degree()* trả về bậc của đa thức.

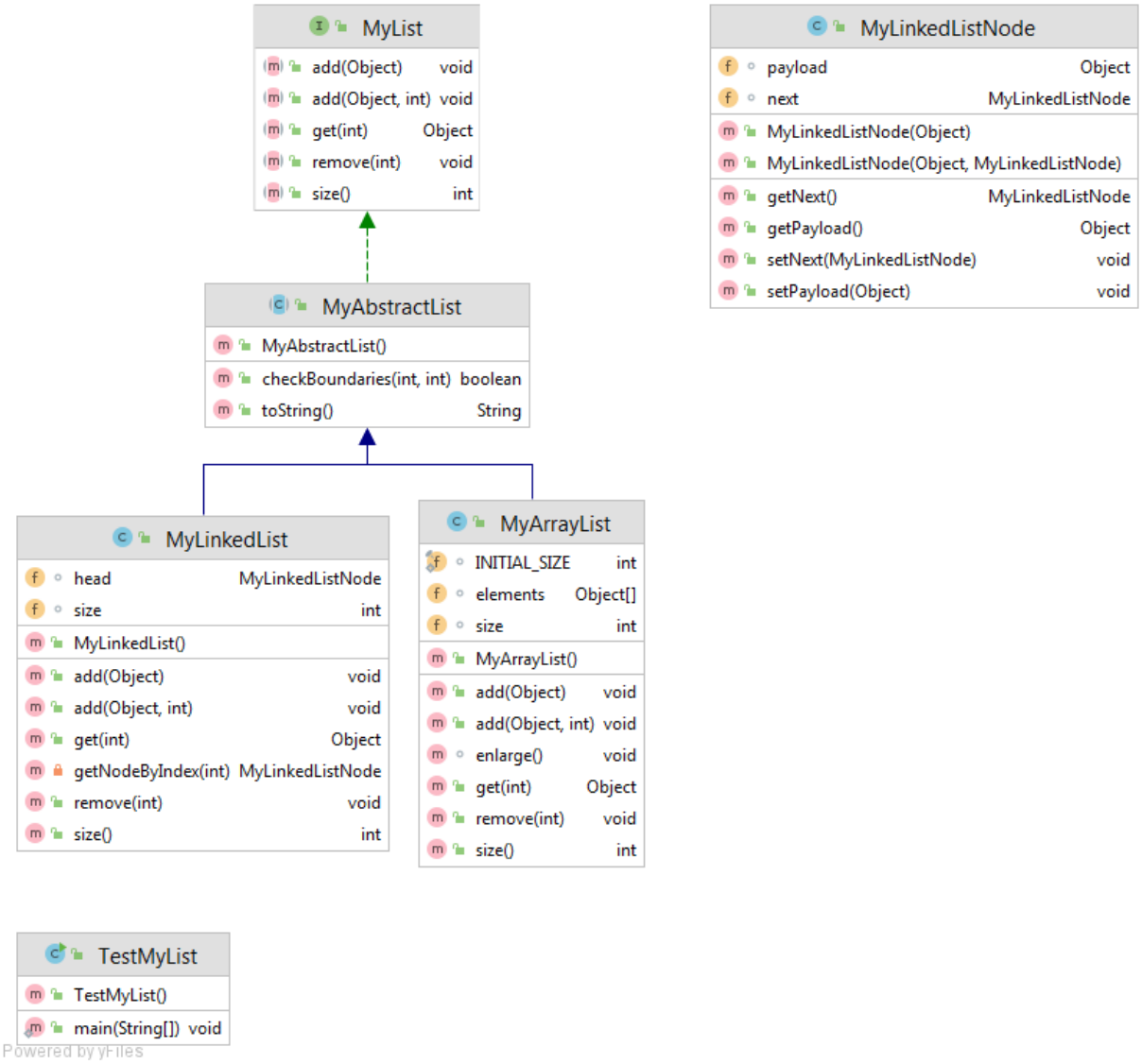
    - Phương thức *derivative()* trả về một đa thức mới là đạo hàm của đa thức ban đầu.
    - Phương thức *evaluate(double x)* tính giá trị của đa thức tại x.
- **AbstractPoly** là một abstract class, có một thuộc tính *degree* là bậc của đa thức.
    - Phương thức *differentiate()* tính đạo hàm của đa thức, trả về mảng là các hệ số của đa thức đạo hàm.
    - Phương thức *equals(Object)* kiểm tra xem hai đa thức có bằng nhau hay không. Hai đa thức bằng nhau nếu cả hai đa thức khác null, có bậc bằng nhau và có các hệ số bằng nhau.
    - Phương thức *toString()* trả về một String mô tả đa thức có định dạng: "Poly[$a_0 + a_1x + a_2x\hat{}2 + ... + a_nx\hat{}n$]".
- **ArrayPoly** và **ListPoly** là hai lớp cài đặt cụ thể cho các đa thức, có dữ liệu được lưu theo kiểu array và List tương ứng. Các hệ số được lưu theo thứ tự từ bậc thấp đến bậc cao.
    - Phương thức *derivative()* tính đạo hàm của đa thức và trả về đa thức đạo hàm. Nếu lấy đạo hàm của đa thức dạng ArrayPoly thì đa thức đạo hàm trả về có dạng ArrayPoly, nếu lấy đạo hàm của đa thức dạng ListPoly thì đa thức đạo hàm trả về có dạng ListPoly.
    - Phương thức *reduce()* tính toán lại xem các hệ số bậc cao nhất của đa thức có bằng 0 hay không, nếu bằng 0 thì điều chỉnh lại *degree* cho thích hợp để không sử dụng đến những phần tử này.
    - Phương thức *type()* trả lại String miêu tả loại đa thức là "Array Poly" hay "List Poly".

Lớp test driver **TestPoly** thực hiện các công việc sau:

- Tạo ra 10 đa thức, trong đó có 5 đa thức loại ArrayPoly và 5 đa thức loại ListPoly, lưu các đa thức này vào List *polyList*.
- In ra các thông tin sau, mỗi thông tin trên một dòng: Đa thức; đạo hàm của đa thức; một giá trị x nào đó; giá trị của đa thức tại x; loại đa thức. Ví dụ

```
Poly 1:
1 + 2x + 3x^2
2 + 6x
3
20
Array Poly
Poly2:
…
```

Câu 3. Cho chương trình Java được thiết kế như biểu đồ dưới đây.



Viết code cho các lớp trong chương trình. Các lớp thuộc package “com.mylist”.

Chú thích:

Các lớp service:

- Chương trình thực hiện cài đặt một kiểu dữ liệu trừu tượng có cấu trúc dữ liệu kiểu List.
- **MyList** là một interface, có các API cho phép người dùng sử dụng List.
- **MyAbstracList** là một abstract class.

    - Phương thức *checkBoundaries(int index, int limit)* kiểm tra xem chỉ số index có nằm trong khoảng [0, limit] hay không.
    - Phương thức *toString()* in ra nội dung List theo định dạng "[phần tử 0] [phần thử 1] …".
- Các lớp **MyLinkedList** và **MyArrayList** cài đặt cụ thể cho các List, sử dụng các cấu trúc dữ liệu khác nhau, kiểu Linked List và Array List.
    - Phương thức *add(Object)* thêm dữ liệu vào cuối List.
    - Phương thức *enlarge()* tạo ra một mảng có kích thước gấp đôi kích thước cũ, và sao chép dữ liệu từ mảng cũ sang mảng mới có kích thước lớn hơn để sử dụng.

Lớp test driver **TestMyList** để test chương trình, thực hiện các công việc sau:

- Tạo ra hai List, một kiểu **MyLinkedList**, một kiểu **MyArrayList**.
- Thực hiện add, remove, get và kiểm tra kích thước dữ liệu trên các List đã tạo (add ít nhất 10 dữ liệu cho mỗi List). In ra dữ liệu có trên List. Mỗi lần add hay remove thì in ra dữ liệu của List trên một dòng. Ví dụ

```
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10]
Add at index 2 value 11:
[1] [2] [11] [3] [4] [5] [6] [7] [8] [9] [10]
Remove at index 5:
[1] [2] [11] [3] [4] [6] [7] [8] [9] [10]
…
```

Chú ý:

- Bài thi có cung cấp các file source code có gợi ý đi kèm. Thí sinh thực hiện bài thi theo các file source code gợi ý. Có thể viết thêm các hàm phụ.
- Thí sinh được phép sử dụng tài liệu và tra cứu internet.
- Thí sinh không được trao đổi bài. Trong thời gian làm bài thi, thí sinh tắt điện thoại, không được mở các ứng dụng chat. Nếu vi phạm sẽ không được tiếp tục làm bài thi và nhận điểm 0.
- Thí sinh nộp bài trên Google Classroom. Thí sinh nộp bài bằng cách nén các file bài làm vào trong file zip, có cả các thư mục tương ứng với các package. Kết quả chạy các chương trình được copy vào 1 file text và nộp cùng các file source code.
