Câu 1: Nêu và giải thích ngắn gọn các đặc tính của ngôn ngữ lập trình hướng đối tượng JAVA.

Câu 2: Cho biết kết quả khi chạy đoạn code sau:

```
public class A {
static int data;
public static void main(String[] args) {
      A a1 = new A();
      a1.data = 100;
      A a2 = new A();
      a2.data = 200;
      int result = a1.data + a2.data;
      System.out.println(a1.data+"+"+a2.data+"="+result); }
}
```

Câu 3: Xây dựng cấu trúc cây nhị phân tìm kiếm (BST) với mỗi node có cấu trúc như sau:

```
public class Student {
int Student_ID;
String Student_name;
double Student_Result;
Student Prev;
Student Next;
public Student(int ID,String name,double result){
      this.Student_ID = ID;
      this.Student_name = name;
      this.Student_Result = result;
}}
```

Giả sử: Student List là 1 danh sách liên kết đối tượng là sinh viên. Viết code cho các phương thức:

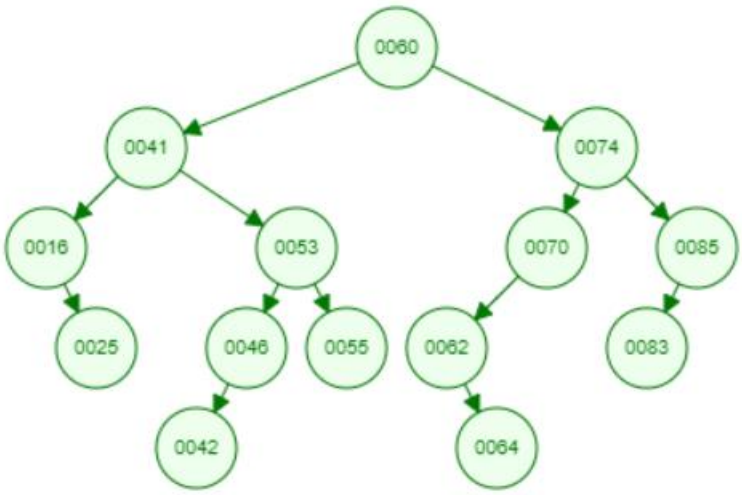
a) Thêm sinh viên vào danh sách
b) Xoá 1 sinh viên khỏi danh sách
c) Xuất thông tin: kết qủa trung bình, sinh viên có kết quả điểm cao nhất, thấp nhất.
d) Thống kê số sinh viên đạt điểm giỏi, khá, trung bình và yếu.

ÔN TẬP CUỐI KÌ LẬP TRÌNH CNTT TRONG JAVA

Câu 1: Trình bày ý nghĩa khai báo Public, Private, Default, Protected với class, method, variable.

Câu 2: Cho thứ tự các Node xây dựng cây nhị phân:
60, 41, 74, 16 53, 85, 25, 46, 55, 83, 70, 42, 62, 64.

a) Vẽ cây BST với các node nhập vào như trên



b) Vẽ lại cây nhị phân sau khi thêm node 51
c) Vẽ lại cây nhị phân sau khi xóa node 41
d) Viết phương thức tìm số node lá của cây
e) Tìm trung bình các node, có bao nhiêu node lớn hơn trung bình, có bao nhiêu node nhỏ hơn trung bình

```
Class Node{
    int Key;
    Node Left;
    Node Right;
};
```

1) Trình bày về NNLT JAVA, các đặc điểm ngôn ngữ lập trình hướng đối tượng
2) Trình bày về ý nghĩa khai báo Public, Private, Default, Protected với class và method, khai báo STATIC với biến và phương thức.
3) Cấu trúc cây nhị phân, cây nhị phân tìm kiếm: vẽ mô tả cây nhị phân, các thao tác trên cây nhị phân tìm kiếm
4) So sánh và sử dụng các cấu trúc Array, ArrayList, LinkedList, Tree, BST

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

KHOA: CNTT

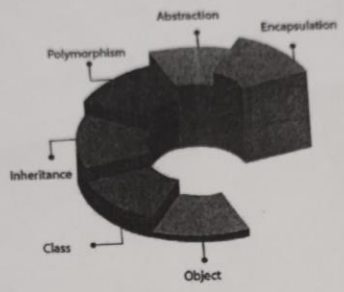
ĐỀ THI CUỐI KỲ

Môn thi : Lập trình JAVA trong CNTT
Lớp/Lớp học phần: DHCNTT18A
Ngày thi: 09/12/2023
Thời gian làm bài:60 phút
(Không kể thời gian phát đề)

Họ và tên thí sinh ....Hồ Phúc Lâm.....; *MSSV* [redacted]

Câu 1: Anh chị hãy giải thích ngắn gọn về Lập trình hướng đối tượng (2đ)

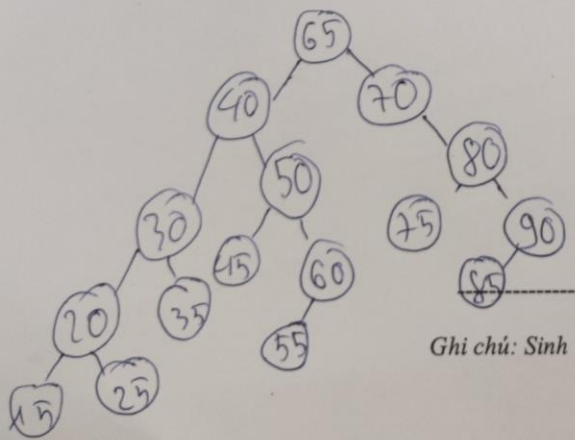
OOPs (Object-Oriented Programming System)

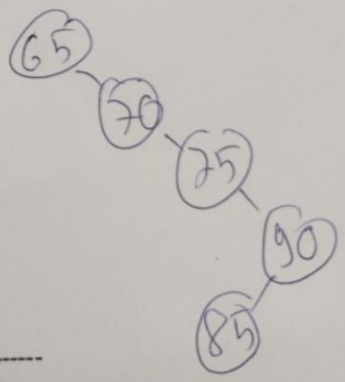


Câu 2: Anh chị hãy giải thích để phân biệt : **instance variable, instance method, static variable, static method** và mô tả **access modifier** của các thành phần trên (2đ)

Câu 3: Cho thứ tự các id các node nhập vào cây nhị phân tìm kiếm:
65 40 50 70 30 80 90 75 60 45 55 35 20 85 15 25

a) Vẽ cây nhị phân tìm kiếm với các node nhập vào như trên (1đ)
b) Duyệt cây nhị phân theo thứ tự LNR, NLR, LRN (2đ)
c) Viết phương thức thực hiện: tìm trung bình tất cả các node, đếm số lượng node lớn hơn, nhỏ hơn trung bình. (2đ)
d) Vẽ lại cây nhị phân tìm kiếm sau khi xoá node 80 (1đ)





---------- Hết ----------

Ghi chú: Sinh viên được sử dụng tài liệu giấy

N L R.
rig ley root.

TRƯỜNG ĐH CÔNG NGHIỆP TP.HCM
PHÒNG KHẢO THÍ & ĐBCL

1

LÝ THUYẾT THI CUỐI KỲ JAVA (tham khảo)

CÂU 1: Trình bày về 4 đặc điểm lập trình hướng đối tượng (Abstraction, Encapsulation, Inheritance và Polymorphism)

Tính đóng gói (encapsulation) :Tức là trạng thái của đối tượng được bảo vệ không cho các truy cập từ code bên ngoài như thay đổi trong thái hay nhìn trực tiếp. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người viết mã. Đây là tính chất đảm bảo sự toàn vẹn, bảo mật của đối tượng Trong Java, tính đóng gói được thể hiện thông qua phạm vi truy cập (access modifier). Ngoài ra, các lớp liên quan đến nhau có thể được gom chung lại thành package.

Tính kế thừa là khả năng cho phép ta xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có. Lớp đã có gọi là lớp Cha, lớp mới phát sinh gọi là lớp Con và đương nhiên kế thừa tất cả các thành phần của lớp Cha, có thể chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.

Tính đa hình (polymorphism):

Khi một tác vụ được thực hiện theo nhiều cách khác nhau được gọi là tính đa hình.

Đối với tính chất này, nó được thể hiện rõ nhất qua việc gọi phương thức của đối tượng. Các phương thức hoàn toàn có thể giống nhau, nhưng việc xử lý luồng có thể khác nhau. Nói cách khác: Tính đa hình cung cấp khả năng cho phép người lập trình gọi trước một phương thức của đối tượng, tuy chưa xác định đối tượng có phương thức muốn gọi hay không. Đến khi thực hiện (run-time), chương trình mới xác định được đối tượng và gọi phương thức tương ứng của đối tượng đó. Kết nối trễ giúp chương trình được uyển chuyển hơn, chỉ yêu cầu đối tượng cung cấp đúng phương thức cần thiết là đủ.

Trong Java, chúng ta sử dụng nạp chồng phương thức (method overloading) và ghi đè phương thức (method overriding) để có tính đa hình.

Tính trừu tượng (abstraction):

Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Tính trừu tượng cho phép bạn loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình.

Tính trừu tượng giúp bạn tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.

Trong Java, chúng là sử dụng abstract class và abstract interface để có tính trừu tượng.

TÍNH QUAN TRỌNG NHẤT LÀ TÍNH : ĐÓNG GÓI

Câu 2: Trình bày các đặc điểm ngôn ngữ lập trình JAVA, lý do chọn JAVA và các khái niệm (JVM, JRE, JDK, SDK, JAVA API, IDE)

-Các đặc điểm ngôn ngữ lập trình JAVA

+Tính trừu tượng (Abstraction): là tiến trình xác định và nhóm các thuộc tính, các hành động liên quan đến một thực thể đặc thù, xét trong mối tương quan với ứng dụng đang phát triển.

+Tính đa hình (Polymorphism): cho phép một phương thức có các tác động khác nhau trên nhiều loại đối tượng khác nhau. Với tính đa hình, nếu cùng một phương thức ứng dụng cho các đối tượng thuộc các lớp khác nhau thì nó đưa đến những kết quả khác nhau. Bản chất của sự việc chính là phương thức này bao gồm cùng một số lượng các tham số.

+Tính kế thừa (Inheritance): điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại.

+Tính đóng gói (Encapsulation): là tiến trình che giấu việc thực thi những chi tiết của một đối tượng đối với người sử dụng đối tượng ấy.

Ngoài ra Java còn có một số đặc điểm sau:

+Độc lập nền (Write Once, Run Anywhere): Không giống như nhiều ngôn ngữ lập trình khác như C và C ++, khi Java được biên dịch, nó không được biên dịch sang mã máy cụ thể, mà thay vào đó là mã bytecode chạy trên máy ảo Java (JVM). Điều này đồng nghĩa với việc bất cứ thiết bị nào có cài đặt JVM sẽ có thể thực thi được các chương trình Java.

+Đơn giản: Học Java thật sự dễ hơn nhiều so với C/C++, nếu bạn đã quen với các ngôn ngữ lập trình hướng đối tượng thì việc học Java sẽ dễ dàng hơn. Java trở nên đơn giản hơn so với C/C++ do đã loại bỏ tính đa kế thừa và phép toán con trỏ từ C/C++.

+Bảo mật: Java hỗ trợ bảo mật rất tốt bởi các thuật toán mã hóa như mã hóa một chiều (one way hashing) hoặc mã hóa công cộng (public key)…

Thông dịch: Java là một ngôn ngữ lập trình vừa biên dịch vừa thông dịch. Chương trình nguồn viết bằng ngôn ngữ lập trình Java có đuôi *.java và được biên dịch thành tập tin có đuôi *.class sau đó được trình thông dịch thông dịch thành mã máy.

Đa luồng: Với tính năng đa luồng Java có thể viết chương trình có thể thực thi nhiều task cùng một lúc. Tính năng này thường được sử dụng rất nhiều trong lập trình game.

Hướng đối tượng: Hướng đối tượng trong Java tương tự như C++ nhưng Java là một ngôn ngữ lập trình hướng đối tượng hoàn toàn. Tất cả mọi thứ đề cập đến trong Java đều liên quan đến các đối tượng được định nghĩa trước, thậm chí hàm chính của một chương trình viết bằng Java (đó là hàm main) cũng phải đặt bên trong một lớp. Hướng đối tượng trong Java không

có tính đa kế thừa (multi inheritance) như trong C++ mà thay vào đó Java đưa ra khái niệm interface để hỗ trợ tính đa kế thừa.
Hiệu suất cao: Nhờ vào trình thu gom rác (garbage collection), giải phóng bộ nhớ đối với các đối tượng không được dùng đến.
Linh hoạt: Java được xem là linh hoạt hơn C/C ++ vì nó được thiết kế để thích ứng với nhiều môi trường phát triển.
*Lý do chọn JAVA
1. Java rất dễ tìm hiểu
2. Java là một ngôn ngữ lập trình hướng đối tượng
3. Số lượng hàm dùng sẵn (API function) của Java hết sức phong phú
4. Các công cụ phát triển mạnh mẽ như Eclipse, Netbeans
5. Bộ sưu tập thư viện mã nguồn mở phong phú
6. Hỗ trợ cộng đồng tuyệt vời
7. Java là miễn phí
8. Hỗ trợ tài liệu xuất sắc – Javadocs
9. Java là nền tảng độc lập
10. Java có mặt ở khắp mọi nơi
*Các khái niệm (JVM, JRE, JDK, SDK, JAVA API, IDE)
JVM: **JVM (Java Virtual Machine)** là máy ảo dùng để chạy chương trình Java. Khi chạy chương trình Java, trình biên dịch Java sẽ biên dịch Java code thành bytecode. Sau đó, JVM sẽ thông dịch bytecode thành mã máy để CPU thực thi.
JRE (Java Runtime Environment) là môi trường thực thi Java bao gồm JVM, các thư viện và những thành phần bổ sung để chạy những ứng dụng được viết bằng Java.
JDK (Java Development Kit) là bộ công cụ phát triển phần mềm Java. JDK (Java Development Kit) bao gồm JRE và những công cụ cần thiết để phát triển ứng dụng bằng ngôn ngữ Java. Khi download JDK thì JRE cũng được tích hợp trong JDK.
SDK là từ viết tắt của Software Development Kit. Nó thực chất là bộ công cụ và phần mềm phục vụ cho việc phát triển ứng dụng dựa trên một nền tảng nhất định.
Cụ thể, SDK cung cấp bộ thư viện, mẫu template, tài liệu, mẫu code, các tiện ích gỡ rối, ghi chú, tài liệu bổ sung… giúp lập trình viên dễ dàng tích hợp vào ứng dụng hay phần mềm. Phần lớn, SDK là chức năng hiển thị thông báo, quảng cáo…
Ngoài ra, SDK còn có thể chứa API được thể hiện dưới dạng thư viện hay một hệ thống phần cứng.
JAVA API: Java xác định cú pháp và ngữ nghĩa của ngôn ngữ lập trình Java. Ngôn ngữ này bao gồm từ vựng và quy tắc cơ bản được sử dụng để viết thuật toán như kiểu dữ liệu nguyên thủy, khối if/else, vòng lặp, v.v.

API là các thành phần phần mềm quan trọng đi kèm với Nền tảng Java. Đây là những chương trình Java viết sẵn có thể "cắm vào là chạy" chức năng hiện tại trong mã của riêng bạn. Ví dụ: Bạn có thể dùng API Java để nhận ngày giờ, thực hiện các phép toán hoặc điều chỉnh văn bản.
Mọi mã ứng dụng Java do nhà phát triển viết ra thường sẽ kết hợp cả mã mới và cũ từ API và thư viện Java.

IDE viết tắt là từ (**I**ntegrated **D**evelopment **E**nvironment) là môi trường tích hợp dùng để viết code để phát triển ứng dụng. Ngoài ra **IDE** tích hợp các tool hỗ trợ khác như trình biên dịch (**Compiler**), trình thông dịch (**Interpreter**), kiểm tra lỗi (**Debugger**), định dạng hoặc highlight code, tổ chức thư mục code, tìm kiếm code…

3. Trình bày về ý nghĩa khai báo Public, Private, Default, Protected với class và method, khai báo STATIC với biến và phương thức. Phân biệt các loại biến: static, instance và local. Các phép toán trên biến.

**Trình bày về ý nghĩa khai báo Public, Private, Default, Protected với class và method:*

-Private, public, protected và default trong Java được gọi chung là Access Modifier có thể được hiểu là công cụ sửa đổi truy cập.

-"Modifier" là 1 từ hoặc cụm từ mô tả có thể thay đổi hoặc sửa đổi nghĩa của 1 từ hoặc cụm từ khác bằng nhiều cách khác nhau. Trong Java, Modifier dùng để đặt mức truy cập cho các class, field, constructor. Access Modifier giúp cập nhật giá trị của 1 biến

-"Access modifier" được dùng trước các khai báo của 1 class, variable hay method thể hiện khả năng truy cập của class, variable hay method đó. Với class thì ta chỉ có 2 loại Access modifier là public và default nhưng với variable và method thì ta có 4 access modifier là public, protected, default, private. Trong đó giá trị của các biến được hiểu như sau:

Private: Chỉ được truy cập trong nội bộ lớp

Public: Có thành phần công khai và có thể truy cập tự do từ bên ngoài

Protected: Có thành phần được bảo vệ, việc truy cập từ bên ngoài bị hạn chế

Default: Chỉ được truy cập trong nội bộ package

1. Private Access Modifier

Trong 4 loại Private, Public, Protected và Default trong Java thì Private Access Modifier là công cụ có phạm vi truy cập mang tính hạn chế nhất. Vì thế các method, variable và constructor nếu được khai báo private thì chỉ có thể được truy cập trong chính lớp khai báo đó.

2. Public Access Modifier

Ngược lại với Private Access Modifier thì Public Access Modifier trong Private, Public, Protected và Default trong Java có thể truy cập được ở bất cứ đâu. Trong tất cả các Modifier thì nó có phạm vi rộng nhất.

3. Protected Access Modifier

Trong 4 loại Private, Public, Protected và Default trong Java thì Protected Access Modifier quy định việc có thể truy cập bên trong package và bên ngoài package chỉ khi thông qua tính kế thừa. Lớp và interface không thể áp dụng được Protected Access Modifier

4. Default Access Modifier

Nếu như chúng ta không khai báo rõ ràng 1 Access Modifier (Private, Public, Protected và Default trong Java) cho một lớp, trường (field), method, ...thì theo mặc định nó được xem là default. Và khi Access Modifier là Default thì các lớp, trường (field), method,... chỉ có thể truy cập được từ bên trong package.

*Khai báo STATIC với biến và phương thức

Biến Static trong Java

Trong Java, các biến tĩnh còn được gọi là các biến lớp. Nếu chúng ta khai báo bất kỳ biến nào với từ khóa static, thì nó được coi là biến tĩnh. Các biến được khai báo với từ khóa static tham chiếu đến thuộc tính chung cho tất cả các đối tượng của lớp. Ví dụ: tên trường đại học giống nhau đối với sinh viên, nhân viên, khoa, v.v. Nói cách khác, chúng ta có thể nói rằng chỉ có một bản sao duy nhất của biến tĩnh ở đó được chia sẻ giữa tất cả các trường hợp của lớp. Chúng ta có thể truy cập trực tiếp các biến tĩnh từ các phương thức tĩnh cũng như không tĩnh.

Phương thức Static trong Java

Phương thức tĩnh là phương thức được khai báo với từ khóa static. Khi chúng ta khai báo một phương thức là static, chúng ta có thể gọi phương thức này hoặc truy cập phương thức này mà không cần tạo một đối tượng hoặc thể hiện của lớp. Như chúng ta biết rằng để gọi các phương thức không tĩnh, trước tiên chúng ta cần tạo đối tượng của lớp và sau đó gọi phương thức thông qua đối tượng, nhưng không giống như các phương thức không tĩnh, chúng ta có thể gọi các phương thức tĩnh trực tiếp với tên lớp.

* Phân biệt các loại biến: static, instance và local:

1. Biến local trong java

Biến local được khai báo trong các phương thức, hàm contructor hoặc trong các block.

Biến local được tạo bên trong các phương thức, contructor, block và sẽ bị phá hủy khi kết thúc các phương thức, contructor và block.

Không được sử dụng "access modifier" khi khai báo biến local.

Các biến local được lưu trên vùng nhớ stack của bộ nhớ.

Bạn cần khởi tạo giá trị mặc định cho biến local trước khi có thể sử dụng.

2. Biến biến instance (biến toàn cục) trong java

Biến instance được khai báo trong một lớp(class), bên ngoài các phương thức, constructor và các block.
Biến instance được lưu trong bộ nhớ heap.
Biến instance được tạo khi một đối tượng được tạo bằng việc sử dụng từ khóa “new” và sẽ bị phá hủy khi đối tượng bị phá hủy.
Biến instance có thể được sử dụng bởi các phương thức, constructor, block, ... Nhưng nó phải được sử dụng thông qua một đối tượng cụ thể.
Bạn được phép sử dụng "access modifier" khi khai báo biến instance, mặc định là "default".
Biến instance có giá trị mặc định phụ thuộc vào kiểu dữ liệu của nó. Ví dụ nếu là kiểu int, short, byte thì giá trị mặc định là 0, kiểu double thì là 0.0d, ... Vì vậy, bạn sẽ không cần khởi tạo giá trị cho biến instance trước khi sử dụng.
Bên trong class mà bạn khai báo biến instance, bạn có thể gọi nó trực tiếp bằng tên khi sử dụng ở khắp nới bên trong class đó.
3. Biến static trong java
Biến static được khai báo trong một class với từ khóa "static", phía bên ngoài các phương thức, constructor và block.
Sẽ chỉ có duy nhất một bản sao của các biến static được tạo ra, dù bạn tạo bao nhiêu đối tượng từ lớp tương ứng.
Biến static được lưu trữ trong bộ nhớ static riêng.
Biến static được tạo khi chương trình bắt đầu chạy và chỉ bị phá hủy khi chương trình dừng.
Giá trị mặc định của biến static phụ thuộc vào kiểu dữ liệu bạn khai báo tương tự biến instance.
Biến static được truy cập thông qua tên của class chứa nó, với cú pháp: TenClass.tenBien.
Trong class, các phương thức sử dụng biến static bằng cách gọi tên của nó khi phương thức đó cũng được khai báo với từ khóa "static".
*Các phép toán trên biến:
Phép gán(=)
*Toán tử toán học(+, -, *, /, %)*
Toán tử tăng, giảm(a++, ++a, a--, --a)
Phép toán quan hệ(>, <, >=, <=, ==, !=)
Phép toán logic(&&, //, !, &, ^)
Phép toán thao tác trên bit:
+Phép toán dịch bit
+Phép toán logic trên bit(AND, OR, XOR, NOT)
*Toán tử gán tắt(+=, -=, *=, /=,...)*
4. Khai báo và sử dụng các cấu trúc Array, ArrayList, LinkedList
*Array

-Để khai báo một mảng, khai báo loại biến với dấu ngoặc vuông []:
Khai báo mảng một chiều
int[] a = **null**;
int b[] = **null**;
Khai báo mảng hai chiều
int[][] a = null;
int b[][] = null;
int c[][] = new int[4][];
-Sử dụng các cấu trúc Array
+ Truy cập các phần tử của một mảng trong Java
+ Thay đổi một phần tử mảng trong Java
+ Độ dài mảng trong Java
+ Duyệt các phần tử của mảng trong Java
+ Sắp xếp mảng
*Array List:
Khởi tạo ArrayList trong java
Có 2 kiểu khởi tạo ArrayList là non-generic và generic:
ArrayList list = **new** ArrayList(); // non-generic-kiểucũ
ArrayList<String> list = **new** ArrayList<String>();
// generic - kiểu mới
-Sử dụng các cấu trúc Array List:
+**Duyệt các phần tử của ArrayList**
+Truy cập phần tử của ArrayList
+Cập nhật giá trị của phần tử Arraylist
+Xóa phần tử ArrayList
+Tìm kiếm một phần tử ArrayList
*Linked List:
Khởi tạo một LinkedList
Để khai báo một LinkedList, chúng ta cần phải import gói thư viện java.util.LinkedList của Java. Cú pháp import như sau:

```
// import gói thư viện java.util.LinkedList
import java.util.LinkedList;
 public class KhoiTaoLinkedList {
    public static void main(String[] args) {
        // khai báo 1 LinkedList có tên là listString
        // có kiểu là String
        LinkedList<String> listString = new LinkedList<String>();
    }
}
```

-Sử dụng các cấu trúc Linked List:
+**Duyệt các phần tử của LinkedList**

+Truy cập phần tử của LinkedList
+Cập nhật giá trị của phần tử Linkedlist
+Xóa phần tử LinkedList
+Tìm kiếm một phần tử LinkedList
Nguồn: https://viettuts.vn/java-collection/linkedlist-trong-java

Câu 5: Cấu trúc cây, cây nhị phân, cây nhị phân tìm kiếm

Cấu trúc dữ liệu phân cấp trong Java

Cấu trúc dữ liệu phân cấp là cấu trúc dữ liệu phi tuyến tính. Các cấu trúc này chủ yếu đại diện cho dữ liệu có chứa mối quan hệ phân cấp giữa các phần tử của nó.

Binary tree - Cây nhị phân trong cấu trúc dữ liệu phân cấp

Binary trees - Cây nhị phân là một cấu trúc trong đó mỗi nút (giao điểm) có thể có nhiều nhất hai nhánh con (hai giao điểm). Trong cây nhị phân, tồn tại một con đường dẫn duy nhất từ nút gốc đến mọi nút khác. Nút trên cùng của cây nhị phân được gọi là nút gốc hoặc nút cha và các nút đền từ nút gốc được gọi là nút con.

Cây nhị phân hoặc rỗng (còn được gọi là cây null) hoặc nó bao gồm một nút gốc cùng với hai nút còn lại, mỗi nút là một cây nhị phân. Mỗi nút trong cây nhị phân có thể có không, 1 hoặc tối đa hai nhánh kế nghiệm hoặc nhánh con tiếp nối. Nhánh cuối được gọi là nhánh lá.

Binary Search Tree (BST)

Binary Search Tree là cấu trúc dữ liệu phân cấp quan trọng nhất khác trong Java. Nó tương tự như cây nhị phân nhưng có một số thuộc tính bổ sung như:

Giá trị của mỗi nhánh N của cây con bên phải lớn hơn mọi giá trị trong cây con bên trái.

Giá trị của mỗi nhánh N của cây con bên trái nhỏ hơn mọi giá trị trong cây con bên phải.

Các cây con bên trái và bên phải là Binary Search Tree.