|  |
| --- |
| **[511643] 자료구조** |
| **실습 #11-12 보고서** |

|  |  |
| --- | --- |
| **이름** | 조아현 |
| **학번** | 20215247 |
| **소속**  **학과/대학** | 소프트웨어융합대학  빅데이터전공 |
| **분반** | 01 (담당교수: 김태운) |

## <주의사항>

* 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
* 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
  + 소스코드/스크립트 등을 작성 한 경우, 해당 파일의 이름도 적어주세요.
* SmartLEAD 제출 데드라인:
  + 3주 뒤 실습시간 전날(다음 다음 다음번 실습 전날) 23:55까지
  + 데드라인을 지나서 제출하면 0점(예외 없음)
  + 주말/휴일/학교행사 등으로 인한 데드라인 연장 없음
  + 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
* SmartLEAD에 아래의 파일을 제출 해 주세요
  + 보고서(PDF 파일로 변환 후 제출 권장하나, WORD 형식으로 제출도 가능)
  + 보고서 파일명에 이름과 학번을 입력 해 주세요.
  + 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일을 하나의 zip 파일로 압축하여 제출(또는 본 문서에 소스코드 화면 캡쳐해서 붙여넣기)

## <개요>

이번 과제는 해시테이블을 구현하는 과제입니다. 총 200점 만점입니다.

*\*\* 참고: 필요한 경우, 강의자료에 나온 코드에서 private을 public으로 변경해서 사용해도 됩니다.*

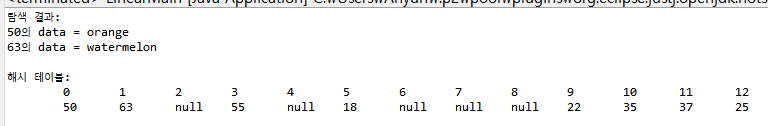
## <실습 과제>

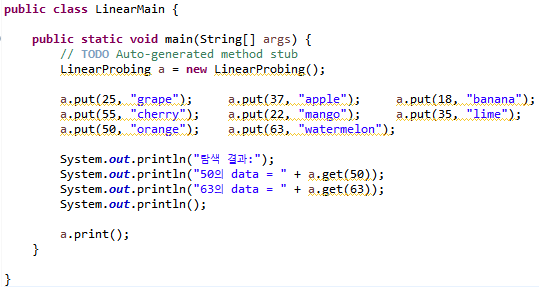
|  |
| --- |
| **[Q 0] 요약 [배점: 10\*2점]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

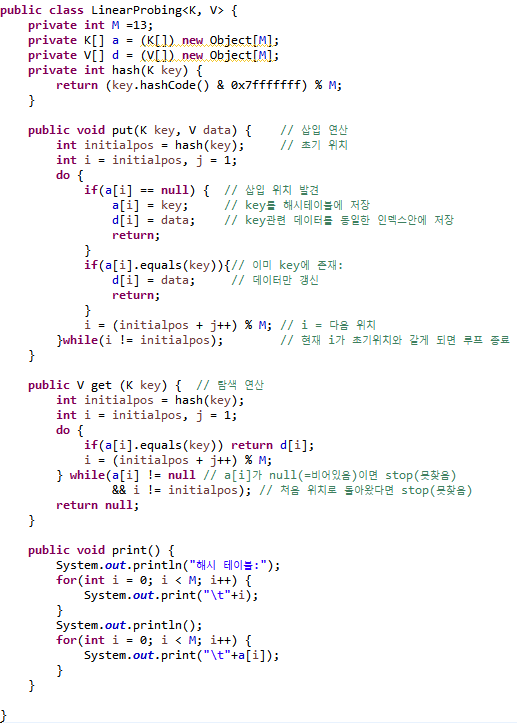
답변: 해시 테이블이 시간 복잡도 성능이 O(logN)보다 좋은 성능을 가지게 할 수 있는 자료구조라는 것을 알게 되었고, 그러기 위해서 ‘키’ 자체를 인덱스로 사용하며 항목을 저장하는 방식을 해싱이라 한다. 여기서 충돌이 발생할 경우 개방주소방식(충돌이 발생한 키를 원래의 해시값과는 다른 곳에 저장)과 폐쇄주소방식(충돌이 발생한 키들은 동일한 해시주소에 저장)으로 구분된다.

|  |
| --- |
| **[Q 1] 선형조사방식, LinearProbing 클래스 [14\*2점]**  강의자료와 동일하게 LinearProbing 클래스를 구현하세요. 강의자료 p. 26과 동일하게 main 메소드를 구현하고 실행하세요. 강의자료에 나온 터미널 출력(= ‘해시 테이블’로 시작하는 출력 부분)과 동일하게 출력하는 print 메소드를 구현하세요. 터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

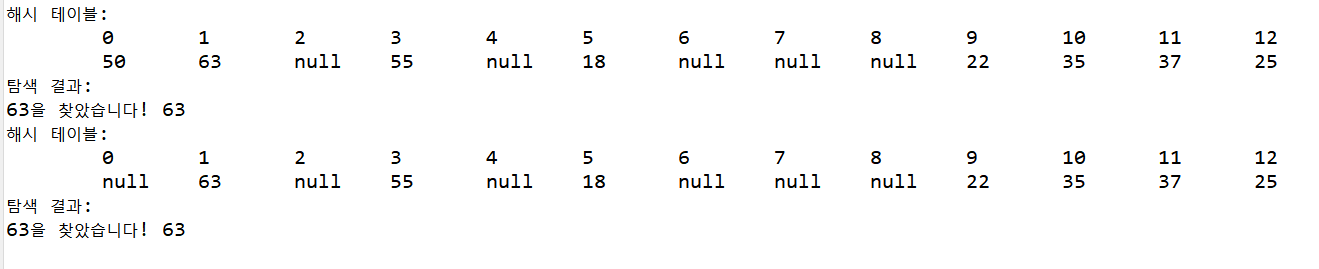
답변:

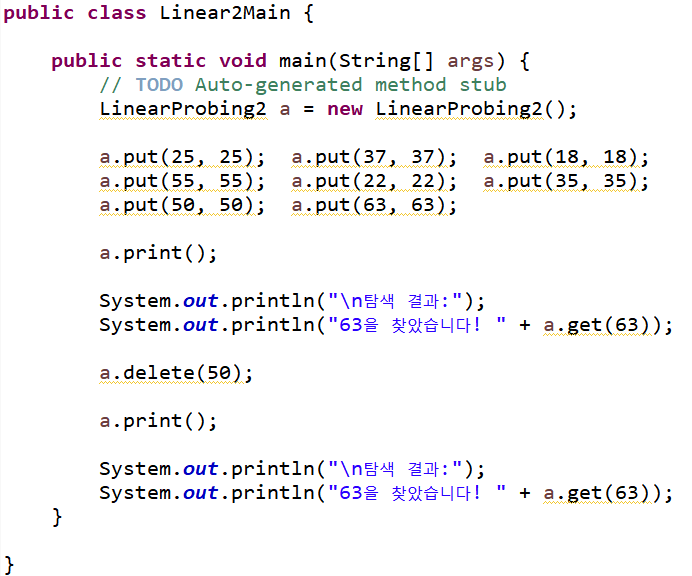




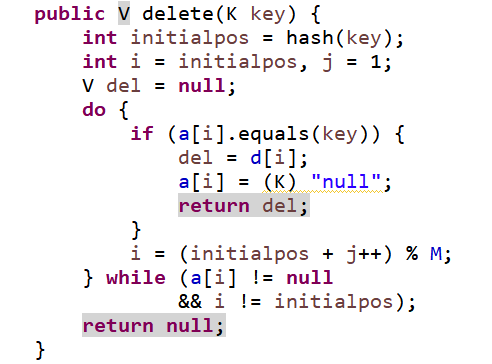


|  |
| --- |
| **[Q 2] LinearProbing2 클래스: 선형조사방식 + delete 메소드 [6\*2점]**  LinearProbing 클래스의 코드를 최대한 재활용해서 LinearProbing2 클래스를 만드세요. LinearProbing2 클래스는 LinearProbing 클래스가 제공하는 모든 기능을 동일하게 제공해야 하며, public V delete(K key) 메소드를 포함하고 있어야 합니다 (키값이 key인 항목을 해시 테이블에서 삭제하고, 삭제된 항목의 데이터를 리턴하도록 구현).   1. 강의자료 p. 27과 같은 순서로 키 값을 추가하세요. K와 V 모두 Integer 형으로 설정하고, put 메소드를 호출할 때, data를 key 값과 같도록 설정하세요. 2. print 메소드를 호출하세요. 3. 키값이 63인 항목을 찾아서 터미널에 출력하세요 (터미널 출력 예시: “63을/를 찾았습니다!”) => 63이 찾아져야 합니다! 4. 키 값이 50인 항목(a[0]에 저장된 항목)을 삭제하세요 5. print 메소드를 호출하세요 6. 키값이 63인 항목(a[1]에 저장된 항목)을 찾아서 터미널에 출력하세요. (터미널 출력 예시: “63을/를 찾았습니다!”) => 63이 찾아져야 합니다!   터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

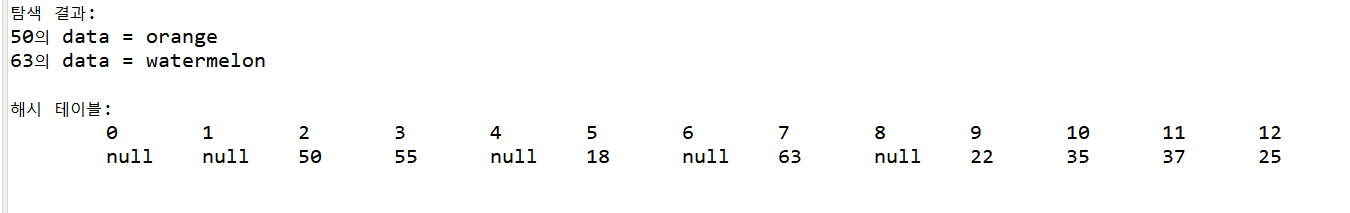
답변: 

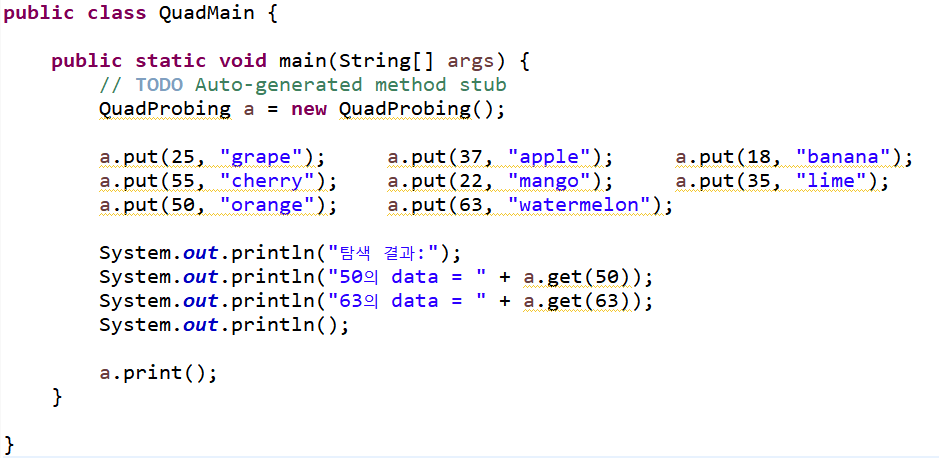


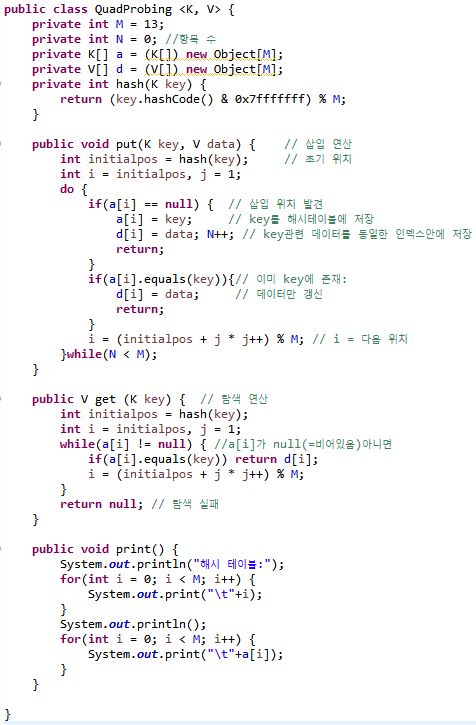




|  |
| --- |
| **[Q3] 이차 조사 방식 [14\*2점]**  강의자료와 동일하게 이차조사방식 QuadProbing 클래스를 구현하세요. 강의자료 p. 32와 같이 테스트를 위한 main 메소드를 구현하고 실행하세요. 강의자료에 나온 터미널 출력(= ‘해시 테이블’로 시작하는 출력 부분)과 동일하게 출력하는 print 메소드를 구현하세요. 터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

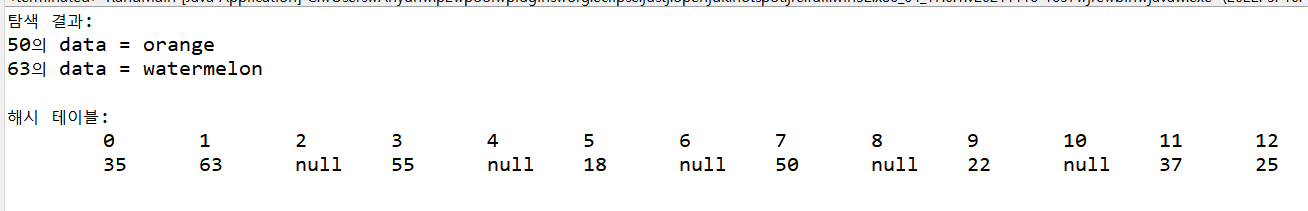
답변: 

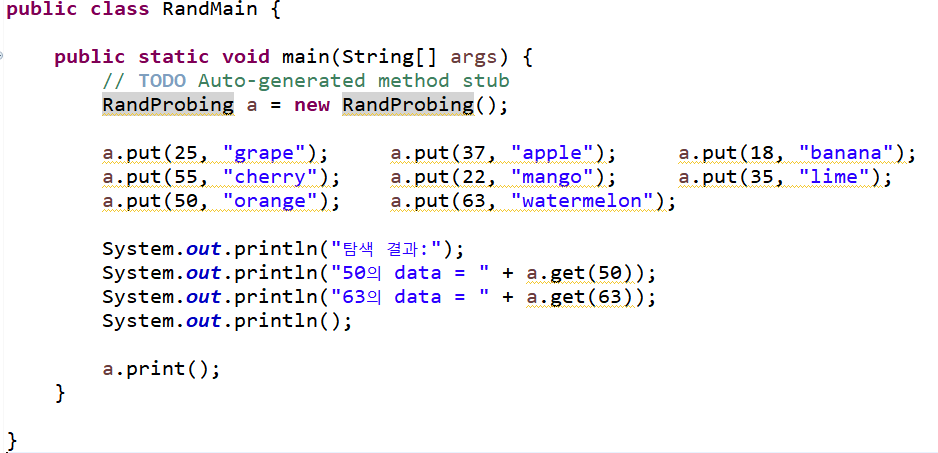


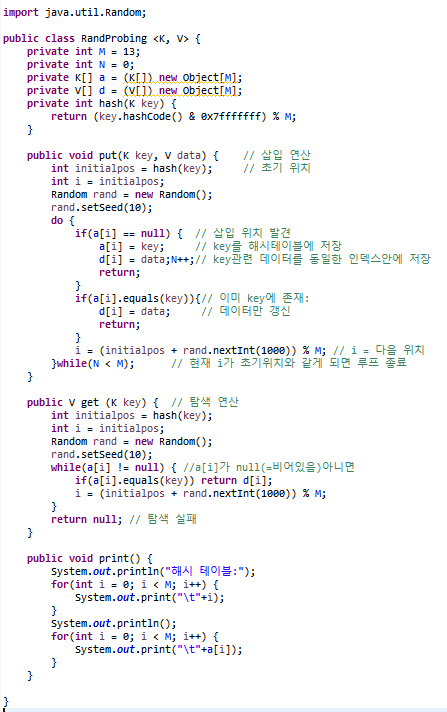


|  |
| --- |
| **[Q4] 랜덤 조사 방식 [14\*2점]**  강의자료와 동일하게 랜덤조사방식 RandProbing 클래스를 구현하세요. 강의자료 p. 35와 같이 테스트를 위한 main 메소드를 구현하고 실행하세요. 강의자료에 나온 터미널 출력(= ‘해시 테이블’로 시작하는 출력 부분)과 동일하게 출력하는 print 메소드를 구현하세요. 터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

답변:

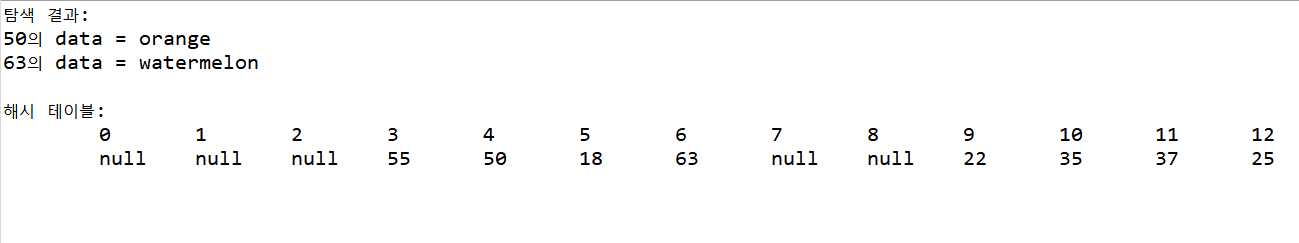
****

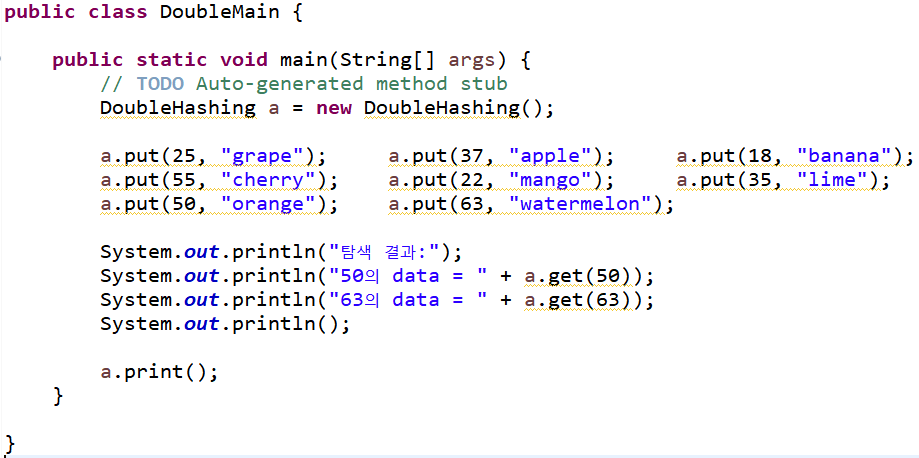


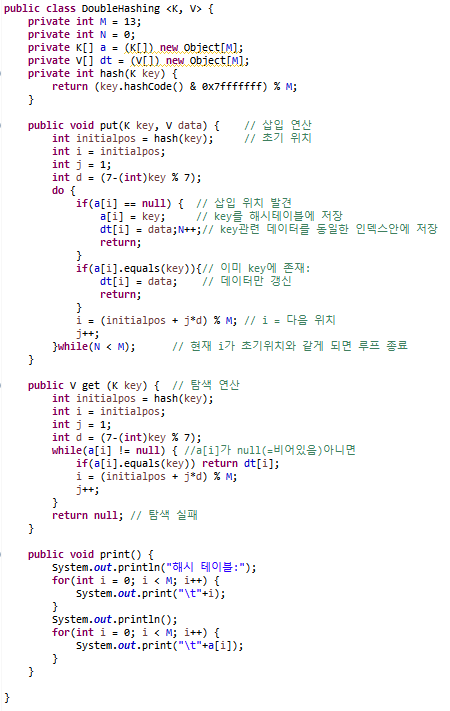


|  |
| --- |
| **[Q5] 이중해싱 [14\*2점]**  강의자료와 동일하게 이중해싱 DoubleHashing 클래스를 구현하세요. 강의자료 p. 40과 같이 테스트를 위한 main 메소드를 구현하고 실행하세요. 강의자료에 나온 터미널 출력(= ‘해시 테이블’로 시작하는 출력 부분)과 동일하게 출력하는 print 메소드를 구현하세요. 터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

답변:

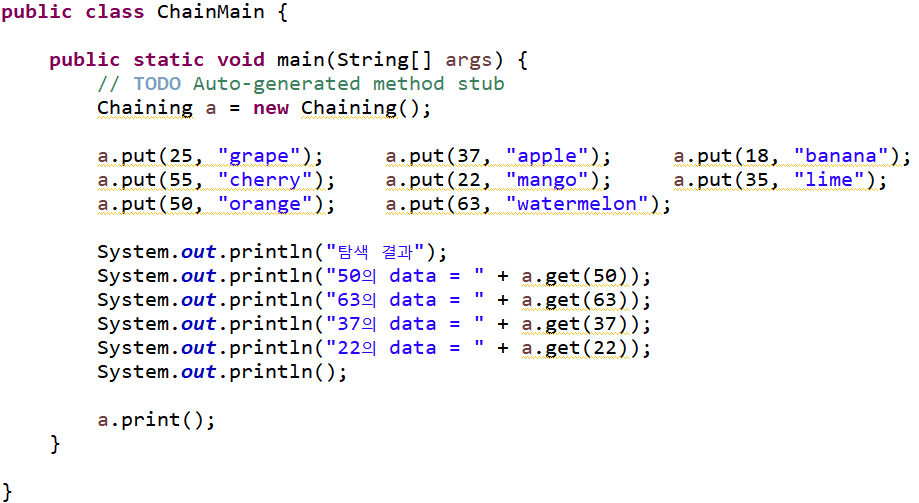
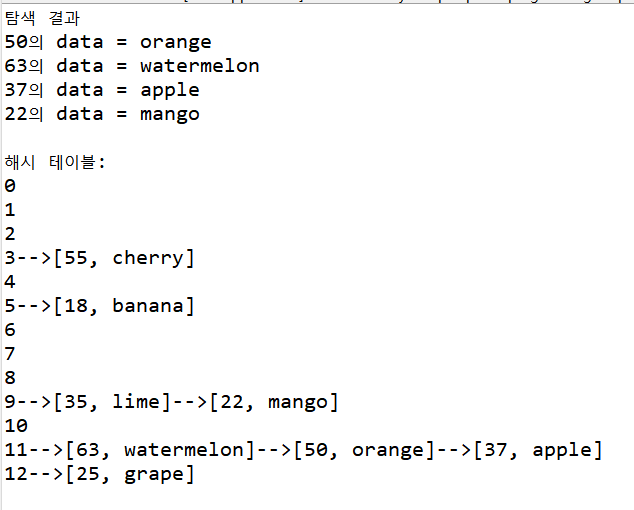
****

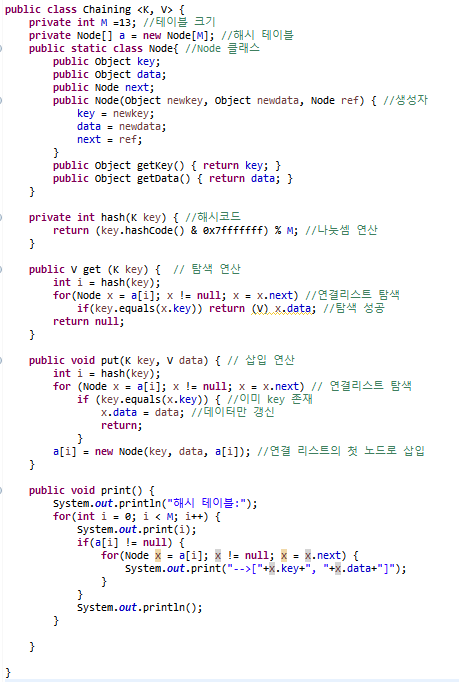
****

****

|  |
| --- |
| **[Q6] 폐쇄주소방식 [14\*2점]**  강의자료와 동일하게 폐쇄주소방식 Chaining 클래스를 구현하세요. 강의자료 p. 47과 같이 테스트를 위한 main 메소드를 구현하세요. Chaining 클래스의 print 메소드는 강의노트 p. 47과 같이 출력하도록 구현해야 합니다 (콘솔 화면 캡처한 이미지에서 ‘해시 테이블:’ 로 시작하는 부분)  터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

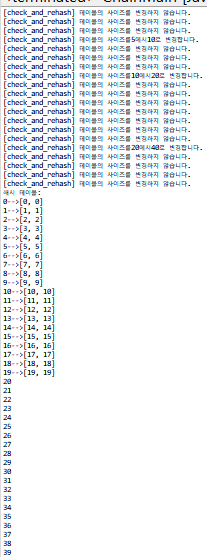
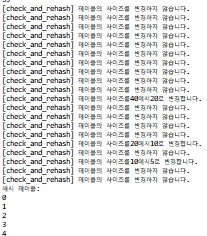
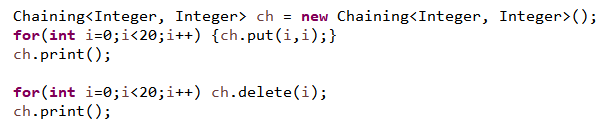
답변:

****

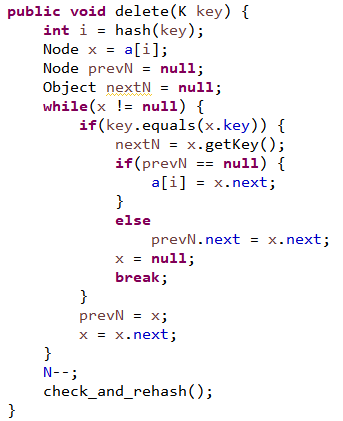
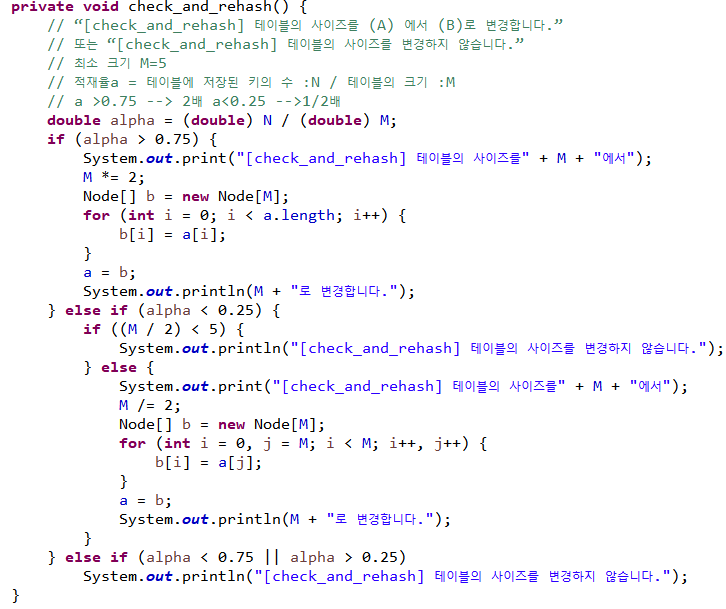
****

|  |
| --- |
| **[Q7] 폐쇄주소방식, 재해시 [6\*2점]**  Chaining 클래스에 재해시를 위한 private void check\_and\_rehash() 메소드를 구현하세요. put 메소드 및 delete 메소드(직접 구현해야 하는 메소드이고, 아래에서 설명)는 아이템을 추가 또는 삭제한 뒤 리턴하기전에 check\_and\_rehash 메소드를 호출하도록 수정하세요.  check\_and\_rehash 메소드는 적재율(강의노트 p. 50 참고)을 계산하고, 적재율이 0.75를 초과하면 해시 테이블을 2배로 늘리고 0.25 이하이면 해시 테이블을 1/2로 줄인 뒤 재해시를 수행하도록 구현하세요. 초기 해시 테이블 크기는 M=5로 설정하세요. 참고로, 해시테이블은 필요한 경우 커지거나 작아질 수 있습니다. 하지만, 해시 테이블의 최소 크기는 M=5이며, 이보다 더 작아질 수는 없습니다. check\_and\_rehash 메소드가 호출될 때 마다 “[check\_and\_rehash] 테이블의 사이즈를 (A) 에서 (B)로 변경합니다.” 또는 “[check\_and\_rehash] 테이블의 사이즈를 변경하지 않습니다.” 라는 메시지를 터미널/콘솔에 출력하도록 구현하세요. 이때, (A)는 현재 테이블 사이즈, (B)는 변경 후 테이블 사이즈입니다.  [Task 1]  아래와 같이 코딩하세요.  Chaining<Integer, Integer> ch = new Chaining<Integer, Integer>();  for(int i=0;i<20;i++) {ch.put(i,i);}  ch.print();  터미널 출력 화면을 캡처해서 본 문서에 첨부하세요.  Chaining 클래스에 public void delete(K key) 메소드를 구현하세요. 이 메소드는 인자로 주어진 key 값을 키 값으로 가지는 노드를 삭제합니다.  [Task 2]  아래와 같이 코딩하세요.  for(int i=0;i<20;i++) ch.delete(i);  ch.print();  터미널 출력 화면을 캡처해서 본 문서에 첨부하세요.  소스코드도 제출하세요. |

답변:

**** **** 

****

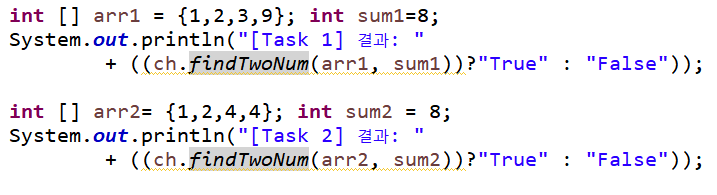
**** ****

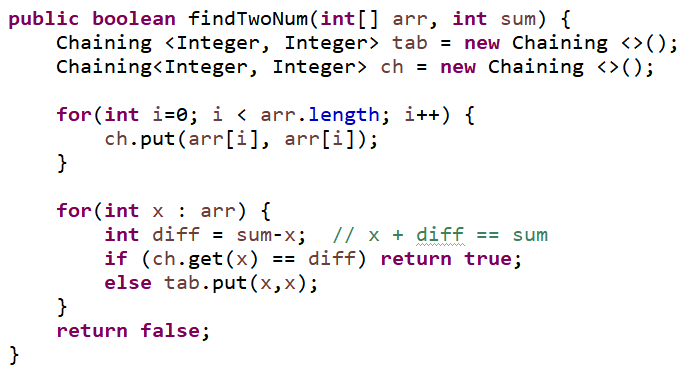
|  |
| --- |
| **[Q8] 창의적 문제해결, 폐쇄주소방식 [8\*2점]**  Chainig 클래스를 사용해서 ‘해시 함수의 활용: 문제’의 ‘솔루션 3’을 구현하시오(p.65). 즉, findTwoNum 메소드를 구현하시오.  [문제 1] arr1=[1,2,3,9], sum1=8 에 대해서 findTwoNum(arr1, sum1)을 호출하고, 메소드의 리턴값을 터미널/콘솔에 다음과 같이 출력하시오: “결과 : False”.  터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요.  [문제 2] arr2=[1,2,4,4], sum2=8 에 대해서 findTwoNum(arr2, sum2)를 호출하고, 메소드의 리턴값을 터미널/콘솔에 다음과 같이 출력하시오: “결과 : True”.  터미널 출력 화면을 캡처해서 본 문서에 첨부하고, 소스코드도 제출하세요. |

답변 1) 

답변 2) 



****

****

**끝! 수고하셨습니다 ☺**