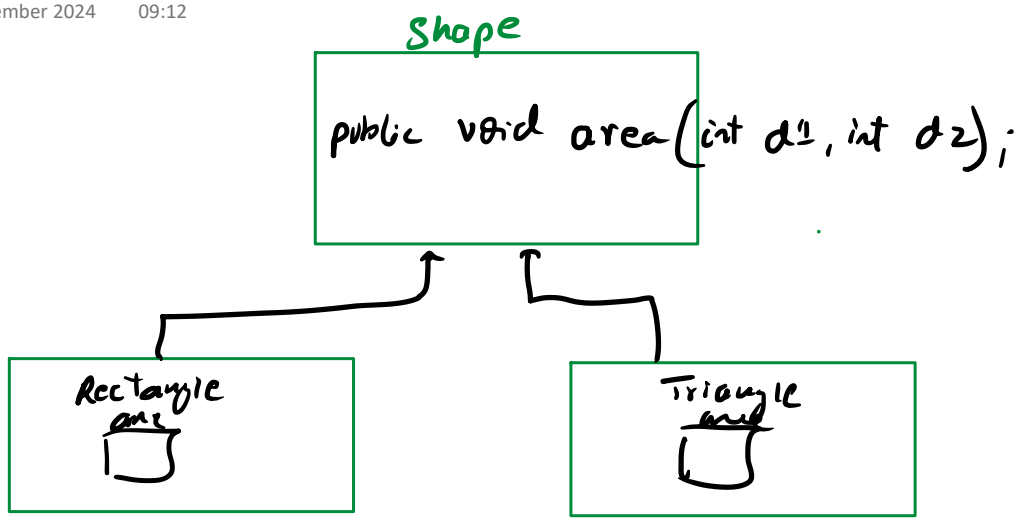
01

21 December 2024 09:12




```
interface Shape
{
    public void area(int d1, int d2);
}
class Rectangle implements Shape
{
    public void area(int l, int b)
    {   float a = l * b;
        S.o.p("Area of rectangle :" + (l*b));
    }
}
class Triangle implements Shape
{
        public void area(int b, int h)
        {
            S.o.p("Area of triangle :" + (0.5f * b * h));
        }
```

```
    }
}

class Main
{
    public static void main(String[] args)
    {
        Shape s;
        s = new Rectangle();
        s.area(10,20);
        s = new Triangle();
        s.area(10,20);
    }
}
```

ArrayList<Employee>

Collection
- boolean add(Employee)
- boolean addAll(Collection<Employee>)
- void remove(Employee)
- boolean contains(Employee)
- boolean isEmpty()
- void clear()
- Iterator<Employee> iterator()

List
- void add(int index, Employee e)
- Employee get(int index)
- void set(int index, Employee e)
- int indexOf(Employee)

class Employee implements Comparable<Employee>
{
public boolean equals(Object obj)
{
Employee e2 = (Employee) obj;
if(this.empno == e2.empno)
return true;
else
return false;
}
public int compareTo(Employee e2)
{
return Integer.compare(this.empno, e2.empno);
}
return this.getName().compareTo(e2.getName());
}

Iterator<Employee> it = list.iterator();
while(it.hasNext())
{
Employee e = it.next();
S.O.P(e);
if(e.getSalary() > 1000)
it.remove();
}

for(Employee e: li
S.O.P(e);

Collections.sort(list)

ArrayList<Integer> list = new
list.add(10);
list.add(20);
list.remove(10);

HashMap
- void put(K,V)
- V get(K)
- Set<K> keySet()
- Collection<V> values()
- V remove(K) → null
- Set<Map.Entry<K,V>> entrySet()

ArrayList<Integer> l = new ArrayList<>(map.values());

LinkedHashSet<String> list = new LinkedHashSet<>(map.keySet());

class NameComparator implements Comparator<Employee>
{
public int compare(Employee e1, Employee e2)
{
return e1.getName().compareTo(e2.getName());
}
}

Collections.sort(list, new NameComparator());

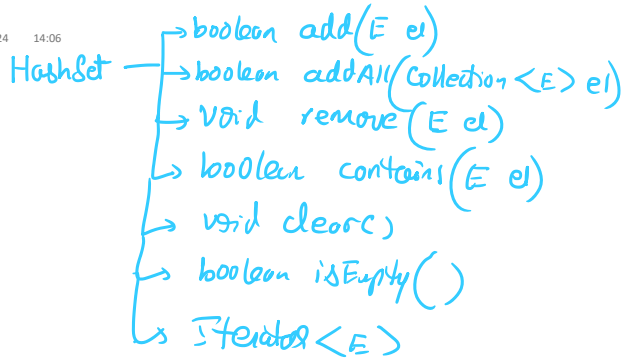
```
int sum=0;
for(Employee e : list)
{
    sum = sum + e.getSalary();
}
```

```
ArrayList<>();
```

```
remove(new Integer(10));
```

03

21 December 2024 14:06



Library Book Borrowing Tracker Using HashMap

You are building a system to track books borrowed from a library. The system should store the titles of books along with the number of times they have been borrowed. Users should be able to add, update, or remove book records, check if a book exists, and retrieve specific borrow counts. Additionally, the system should calculate the total number of books borrowed from the library.

If any operation attempts to access a book that does not exist in the system, a custom exception, BookNotFoundException, should be thrown and handled appropriately.

Operations to be Performed
1.Add Book Records: Input the number of books and their respective titles and borrow counts.

2.Retrieve Borrow Count: Look up the total number of times a specific book has been borrowed by its title.

3.Remove Book Records: Remove a book's record from the system.

4.Check if a Book Exists: Verify if a specific book is in the system based on its title.

5.Calculate Total Borrow Count: Display the total number of times books have been borrowed.

Input Format

Adding Books:
•The first line contains an integer n, representing the number of books to add. If n value is not greater then 0, terminate the program with message "Invalid number".

•For each book, input:
  O The first line contains the book’s title (String).
  O The second line contains the borrow count (Integer).

Perform Operations:

•A line containing the title of a book to retrieve its borrow count.
•A line containing the title of a book to remove from the system.
•A line containing the title of a book to check if it exists in the system.

Output Format

Retrieving Borrow Count:
  •If found, print: [Book title] has been borrowed [count] times.

  •If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found.
  Removing a Book:

  •If found and removed, print: Book [Book title] has been removed from the library.

  •If not found, throw and catch the BookNotFoundException and print: Book [Book title] not found for removal.

Cecking if a Book Exists:
  •If the book exists, print: Book [Book title] exists in the library.

  •If the book doesn’t exist, throw and catch the BookNotFoundException and print: Book [Book title] does not exist in the library.

Calculating Total Borrow Count:
  •Print the total borrow count of all books: Total Books Borrowed: [total count]

Sample Input 1
3

The Alchemist
150
To Kill a Mockingbird
200
1984
100
The Alchemist
To Kill a Mockingbird
1984

Sample Output 1
The Alchemist has been borrowed 150 times.
Book To Kill a Mockingbird has been removed from the library.
Book 1984 exists in the library.
Total Books Borrowed: 250

Sample Input 2
2
The Great Gatsby
250
Pride and Prejudice
300
Moby Dick
War and Peace
Anna Karenina

Sample Output 2
Book Moby Dick not found.
Book War and Peace not found for removal.
Book Anna Karenina does not exist in the library.
Total Books Borrowed: 550

```
String borrowBook = scan.nextLine();
if(map.containsKey(borrowBook))
{
    int count = map.get(borrowBook);
    count++;
    if(count > 100)
    {
        map.remove(borrowBook);
        s.o.p(borrowBook + "is removed");
    }
    map.put(borrowBook, count);
    s.o.p("borrowBook +" borrowed" + count + "times");
}
else
```

Search the book that starts with given data

```
int cnt=0;
String startString = scan.nextLine();
for(Map.Entry<String, Integer> e : map.entrySet())
{
    if(e.getKey().startsWith(startString))
    {
        cnt++;
```

String
startsWith
endsWith
contains

```
if (e.getKey().startsWith(startString))
{
    cnt++;
    S.O.P(e);
}
}
S.O.P(cnt);
```

endswith
contains
S.indexOf("str")

HashSet with custom object

HashSet<Task> → equals, hashCode

```
int cnt = 0;
Iterator<Campaign> it = set.iterator();
while(it.hasNext())
{
    Campaign c = it.next();
    if(c.getBudget() < 5000)
        it.remove
        cnt++;
}
S.o.p("No of campaigns deleted :"+cnt);
```

A

```
String data = scan.nextLine();
"SELECT * from student WHERE studentname LIKE '" + data + "%'"
"SELECT * FROM student WHERE studentName LIKE '%" + data + "'"
d1 = scan.nextLine();
d2 = scan.nextLine();
"SELECT * FROM student WHERE studentname LIKE '" + d1 + "%'
```

LIKE '%A'

(studentname LIKE 'A%') AND (studentName LIKE '%H')

studentname like '%A'

AND studentname LIKE '%" + dz + "')