1. **import** java.util.\*;
3. **class** Repo {
4. **private** **static** ArrayList<String> missions = **new** ArrayList<String>();
5. **private** **static** **boolean** k = **true**;
7. **public** Repo(String items) {
8. String[] itemss = items.split("\\s+");
9. **for**(String item : itemss)
10. missions.add(item);
11. }
13. **int** getSize() {
14. **return** missions.size();
15. }
17. **public** **synchronized** **void** give1() {
18. **if**(k == **false**) {
19. **try** {
20. **this**.wait();
21. }
22. **catch**(Exception e) {
23. e.printStackTrace();
24. }
25. }
26. **else** {
27. System.out.println(Thread.currentThread().getName() + " finish " + missions.get(0));
28. missions.remove(0);
29. k = **false**;
30. **this**.notify();
31. }
32. }
34. **public** **synchronized** **void** give2() {
35. **if**(k == **true**) {
36. **try** {
37. **this**.wait();
38. }
39. **catch**(Exception e) {
40. e.printStackTrace();
41. }
42. }
43. **else** {
44. System.out.println(Thread.currentThread().getName()+ " finish " + missions.get(0));
45. missions.remove(0);
46. k = **true**;
47. **this**.notify();
48. }
49. }
50. }
52. **class** Worker1 **implements** Runnable {
53. Repo repo;
54. Worker1(Repo repo) {
55. **this**.repo = repo;
56. }
58. **public** **void** run() {
59. **while**(**true**) {
60. **if**(repo.getSize() == 0) **break**;
61. repo.give1();
62. }
63. }
64. }
66. **class** Worker2 **implements** Runnable {
67. Repo repo;
68. Worker2(Repo repo) {
69. **this**.repo = repo;
70. }
72. **public** **void** run() {
73. **while**(**true**) {
74. **if**(repo.getSize() == 0) **break**;
75. repo.give2();
76. }
77. }
78. }
79. **import** java.util.\*;
81. **class** Student{
82. **private** String name;
83. **private** String sex;
84. **private** **int** age;
86. @Override
87. **public** String toString() {
88. String str="Student [name='"+name+"', sex='"+sex+"', age="+age+"]";
89. **return** str;
90. }
92. **public** Student(String name, String sex, **int** age) {
93. **this**.name = name;
94. **this**.sex = sex;
95. **this**.age = age;
96. }
98. **public** String getName() {
99. **return** name;
100. }
102. **public** **void** setName(String name) {
103. **this**.name = name;
104. }
106. **public** String getSex() {
107. **return** sex;
108. }
110. **public** **void** setSex(String sex) {
111. **this**.sex = sex;
112. }
114. **public** **int** getAge() {
115. **return** age;
116. }
118. **public** **void** setAge(**int** age) {
119. **this**.age = age;
120. }
121. }
122. **public** **class** Main {
123. **public** **static** **void** main(String[] args) {
124. Scanner sc=**new** Scanner(System.in);
125. String name=sc.next();
126. **int** age=Integer.valueOf(sc.next());
127. String sx=sc.next();
128. Student student = **new** Student(name, sx, age);
129. System.out.println(student);
130. }
131. }
132. **import** java.util.\*;
133. **public** **class** Main {
135. **public** **static** **void** main(String[] args) {
136. // TODO Auto-generated method stub
137. Scanner sc = **new** Scanner(System.in);
138. **int** []a = **new** **int** [5];
139. **while**(**true**)
140. {
141. String first;
142. first = sc.next();
143. **if**(first.equals("other"))
144. **break**;
145. **if**(first.equals("arr"))
146. {
147. **try** {
148. **int** secend = sc.nextInt();
149. **int** t = a[secend];
150. }**catch**(Exception e){
151. System.out.println(e);
152. }
153. }
154. **if**(first.equals("null"))
155. {
156. **try** {
157. String t = **null**;
158. **int** length = t.length();
159. }**catch**(Exception e) {
160. System.out.println(e);
161. }
162. }
163. **if**(first.equals("cast"))
164. {
165. **try** {
166. Object ss = **new** String("string");
167. //Integer t = (Integer)ss;
168. System.out.println((Integer)ss);
169. }**catch**(Exception e){
170. System.out.println(e);
171. }
172. }
173. **if**(first.equals("num"))
174. {
175. **try** {
176. String c = sc.next();
177. Integer num = Integer.parseInt(c);
178. //System.out.println(Integer.parseInt(c));
179. }**catch**(Exception e)
180. {
181. System.out.println(e);
182. }
183. }
185. }
186. sc.close();
187. }
188. }
189. **import** java.util.\*;
191. **class** IllegalScoreException **extends** Exception{
192. **public** IllegalScoreException() {
193. }
195. **public** IllegalScoreException(String message) {
196. **super**(message);
197. }
199. **public** IllegalScoreException(String message, Throwable cause) {
200. **super**(message, cause);
201. }
203. **public** IllegalScoreException(Throwable cause) {
204. **super**(cause);
205. }
206. }
208. **class** IllegalNameException **extends** Exception{
209. **public** IllegalNameException() {
210. }
212. **public** IllegalNameException(String message) {
213. **super**(message);
214. }
216. **public** IllegalNameException(String message, Throwable cause) {
217. **super**(message, cause);
218. }
220. **public** IllegalNameException(Throwable cause) {
221. **super**(cause);
222. }
223. }
225. **class** Student{
226. **private** String name;
227. **private** **int** score;
228. //一般情况下，需要注意若程序抛出一个异常后，程序停止执行
229. **private** **int** flag=0;
230. **public** String getName() {
231. **return** name;
232. }
234. **public** **int** getFlag() {
235. **return** flag;
236. }
238. **public** **void** setFlag(**int** flag) {
239. **this**.flag = flag;
240. }
242. **public** **void** setName(String name) {
243. **char** c=name.charAt(0);
244. **if** (c>='0'&&c<='9'){
245. **try** {
246. flag=1;
247. **throw** **new** IllegalNameException();
248. } **catch** (IllegalNameException e) {
249. System.out.println("IllegalNameException: the first char of name must not be digit, name="+name);
250. }
252. **return**;
253. }
254. **this**.name = name;
255. }
257. **public** **int** getScore() {
258. **return** score;
259. }
261. **public** **void** setScore(**int** score)  {
262. **if** (flag==1){
263. **return**;
264. }
265. **if** (score<0||score>100){
266. **try** {
267. flag=1;
268. **throw** **new** IllegalScoreException();
269. } **catch** (IllegalScoreException e) {
270. System.out.println("IllegalScoreException: score out of range, score="+score);
271. }
272. }
273. **this**.score = score;
274. }
275. //如果加分后分数<0 或>100，则抛出IllegalScoreException，加分不成功。
276. **public** **int** addScore(**int** score) {
277. **return** 0;
278. }
280. @Override
281. **public** String toString() {
282. **return** "Student [" +
283. "name=" + name  +
284. ", score=" + score +
285. ']';
286. }
287. }
289. **public** **class** Main {
290. **public** **static** **void** main(String[] args){
291. Scanner sc=**new** Scanner(System.in);
292. **while** (**true**){
293. String str=sc.nextLine();
294. **if** (str.equals("new")){
295. String s=sc.nextLine();
296. String[] s1 = s.split(" ");
297. **if** (s1.length==2){
298. String name=s1[0];
299. **int** score=Integer.valueOf(s1[1]);
300. Student student = **new** Student();
301. student.setName(name);
302. student.setScore(score);
303. **if** (student.getFlag()==0){
304. System.out.println(student);
305. }
306. }**else**{
307. System.out.println("java.util.NoSuchElementException");
308. }
310. }**else** {
311. **break**;
312. }
313. }
314. sc.close();
315. System.out.println("scanner closed");
316. }
317. }
318. **class** WorkerList{
319. Worker readInWorker() {
320. Scanner sc=**new** Scanner(System.in);
321. Worker a= **new** Worker();
322. String nam=sc.next();
323. **double** sal=sc.nextDouble();
324. a.setSalary(sal);
325. a.setName(nam);
326. sc.close();
327. **return** a;
328. }
329. List<Worker> constructWorkerList()
330. {
331. Scanner sc=**new** Scanner(System.in);
332. List<Worker> list=**new** ArrayList<Worker>();
333. **int** n=sc.nextInt();
334. //Worker w=new Worker();
336. String nc;
337. **double** sa;
338. **for**(**int** i=0;i<n;i++)
339. {
340. Worker w=**new** Worker();
341. //w=readInWorker();
342. nc=sc.next();
343. sa=sc.nextDouble();
344. w.setName(nc);
345. w.setSalary(sa);
346. list.add(w);
348. }
349. sc.close();
350. **return** list;
351. }
352. **double** computeTotalSalary(List<Worker> list)
353. {
354. **double** sum=0;
355. **for**(**int** i=0;i<list.size();i++)
356. {
357. sum=sum+list.get(i).getSalary();
358. }
359. **return** sum;
360. }
361. }
362. **public** **static** List<String> convertStringToList(String line)
363. {
364. String[] ss = line.split("\\s+");//String的split方法支持正则表达式
365. List<String> list = **new** ArrayList<String>(Arrays.asList(ss));//将数组转化为list
366. **return** list;
367. }
368. **public** **static** **void** remove(List<String> list, String str)
369. {
370. **if**(!list.contains(str))
371. **return**;
372. **else**
373. {
374. **for**(**int** i = 0;i < list.size();i++)
375. {
376. **if**(str.contains(list.get(i)))
377. {
378. list.remove(list.get(i));
379. i--;//这是难度，删除的话必须加这句，否则你可能会出现跳着删除的情况，如样例一
380. }
382. }
383. }
384. }
385. **import** java.util.ArrayList;
386. **import** java.util.Scanner;
388. **interface** GeneralStack<T>{
389. **public** T push(T item);            //如item为null，则不入栈直接返回null。
390. **public** T pop();                 //出栈，如为栈为空，则返回null。
391. **public** T peek();                //获得栈顶元素，如为空，则返回null.
392. **public** **boolean** empty();//如为空返回true
393. **public** **int** size();     //返回栈中元素数量
394. }
395. **class** ArrayListGeneralStack **implements** GeneralStack{
396. ArrayList list=**new** ArrayList();
397. @Override
398. **public** String toString() {
399. **return**  list.toString();
400. }
402. @Override
403. **public** Object push(Object item) {
404. **if** (list.add(item)){
405. **return** item;
406. }**else** {
407. **return** **false**;
408. }
409. }
411. @Override
412. **public** Object pop() {
413. **if** (list.size()==0){
414. **return** **null**;
415. }
416. **return** list.remove(list.size()-1);
417. }
419. @Override
420. **public** Object peek() {
421. **return** list.get(list.size()-1);
422. }
424. @Override
425. **public** **boolean** empty() {
426. **if** (list.size()==0){
427. **return** **true**;
428. }**else** {
429. **return** **false**;
430. }
431. }
433. @Override
434. **public** **int** size() {
435. **return** list.size();
436. }
437. }
438. **class** Car{
439. **private** **int** id;
440. **private** String name;
442. @Override
443. **public** String toString() {
444. **return** "Car [" +
445. "id=" + id +
446. ", name=" + name  +
447. ']';
448. }
450. **public** **int** getId() {
451. **return** id;
452. }
454. **public** **void** setId(**int** id) {
455. **this**.id = id;
456. }
458. **public** String getName() {
459. **return** name;
460. }
462. **public** **void** setName(String name) {
463. **this**.name = name;
464. }
466. **public** Car(**int** id, String name) {
467. **this**.id = id;
468. **this**.name = name;
469. }
470. }
471. **public** **class** Main {
472. **public** **static** **void** main(String[] args) {
473. Scanner sc=**new** Scanner(System.in);
474. **while** (**true**){
475. String s=sc.nextLine();
476. **if** (s.equals("Double")){
477. System.out.println("Double Test");
478. **int** count=sc.nextInt();
479. **int** pop\_time=sc.nextInt();
480. ArrayListGeneralStack arrayListGeneralStack = **new** ArrayListGeneralStack();
481. **for** (**int** i=0;i<count;i++){
482. System.out.println("push:"+arrayListGeneralStack.push(sc.nextDouble()));
483. }
484. **for** (**int** i=0;i<pop\_time;i++){
485. System.out.println("pop:"+arrayListGeneralStack.pop());
486. }
487. System.out.println(arrayListGeneralStack.toString());
488. **double** sum=0;
489. **int** size=arrayListGeneralStack.size();
490. **for** (**int** i=0;i<size;i++){
491. sum+=(**double**)arrayListGeneralStack.pop();
492. }
493. System.out.println("sum="+sum);
494. System.out.println("interface GeneralStack");
495. }**else** **if** (s.equals("Integer")){
496. System.out.println("Integer Test");
497. **int** count=sc.nextInt();
498. **int** pop\_time=sc.nextInt();
499. ArrayListGeneralStack arrayListGeneralStack = **new** ArrayListGeneralStack();
500. **for** (**int** i=0;i<count;i++){
501. System.out.println("push:"+arrayListGeneralStack.push(sc.nextInt()));
502. }
503. **for** (**int** i=0;i<pop\_time;i++){
504. System.out.println("pop:"+arrayListGeneralStack.pop());
505. }
506. System.out.println(arrayListGeneralStack.toString());
507. **int** sum=0;
508. **int** size=arrayListGeneralStack.size();
509. **for** (**int** i=0;i<size;i++){
510. sum+=(**int**)arrayListGeneralStack.pop();
511. }
512. System.out.println("sum="+sum);
513. System.out.println("interface GeneralStack");
514. }**else** **if** (s.equals("Car")){
515. System.out.println("Car Test");
516. **int** count=sc.nextInt();
517. **int** pop\_time=sc.nextInt();
518. ArrayListGeneralStack arrayListGeneralStack = **new** ArrayListGeneralStack();
519. **for** (**int** i=0;i<count;i++){
520. **int** id=sc.nextInt();
521. String name=sc.next();
522. Car car = **new** Car(id,name);
523. System.out.println("push:"+arrayListGeneralStack.push(car));
524. }
525. **for** (**int** i=0;i<pop\_time;i++){
526. System.out.println("pop:"+arrayListGeneralStack.pop());
527. }
528. System.out.println(arrayListGeneralStack.toString());
529. **if** (arrayListGeneralStack.size()>0){
530. **int** size=arrayListGeneralStack.size();
531. **for** (**int** i=0;i<size;i++){
532. Car car=(Car) arrayListGeneralStack.pop();
533. System.out.println(car.getName());
534. }
535. }
536. System.out.println("interface GeneralStack");
537. }**else** **if** (s.equals("quit")){
538. **break**;
539. }
540. }
542. }
543. }