```
/**¶
* Copyright (c) 2009 ISP RAS.¶
* 109004, A. Solzhenitsina, 25, Moscow, Russia.¶
* All rights reserved.¶
* <p>¶
* $Id$¶
* Created on Jan 13, 2016¶
*/¶
¶
package dashbah.testinghw4;¶
¶
/**¶
* @author Victor Kuliamin¶
*/¶
public class VendingMachine {¶
private long id = 117345294655382L;¶
¶
public enum Mode {OPERATION, ADMINISTERING}¶
¶
;¶
private Mode mode = Mode.OPERATION;¶
¶
public enum Response {¶
OK, ILLEGAL_OPERATION, INVALID_PARAM, CANNOT_PERFORM,
TOO_BIG_CHANGE, UNSUITABLE_CHANGE, INSUFFICIENT_PRODUCT,
INSUFFICIENT_MONEY¶
}¶
¶
;¶
¶
private int max1 = 30;¶
private int max2 = 40;¶
¶
private int num1 = 0;¶
private int num2 = 0;¶
¶
private int price1 = 8;¶
private int price2 = 5;¶
¶
private int maxc1 = 50;¶
private int maxc2 = 50;¶
¶
private int coins1 = 0;¶
private int coins2 = 0;¶
¶
public static int coinval1 = 1;¶
public static int coinval2 = 2;¶
¶
private int balance = 0;¶
¶
public int getNumberOfProduct1() {¶
return num1;¶
}¶
¶
public int getNumberOfProduct2() {¶
return num2;¶
}¶
¶
public int getCurrentBalance() {¶
return balance;¶
}¶
¶
public Mode getCurrentMode() {¶
return mode;¶
}¶
¶
public int getCurrentSum() {¶
if (mode == Mode.OPERATION)¶
return 0;¶
else¶
return coins1 * coinval1 + coins2 * coinval2;¶
}¶
¶
public int getCoins1() {¶
if (mode == Mode.OPERATION)¶
return 0;¶
else¶
return coins1;¶
}¶
¶
public int getCoins2() {¶
if (mode == Mode.OPERATION)¶
return 0;¶
else¶
return coins2;¶
}¶
¶
public int getPrice1() {¶
return price1;¶
}¶
¶
public int getPrice2() {¶
```

```
return price2;¶
}¶
¶
public Response fillProducts() {¶
if (mode != Mode.ADMINISTERING) {¶
return Response.ILLEGAL_OPERATION;¶
}¶
num1 = max1;¶
num2 = max2;¶
return Response.OK;¶
}¶
¶
public Response fillCoins(int c1, int c2) {¶
if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;¶
if (c1 <= 0 || c1 > maxc1) return Response.INVALID_PARAM;¶
if (c2 <= 0 || c2 > maxc2) return Response.INVALID_PARAM;¶
coins1 = c1;¶
coins2 = c2;¶
return Response.OK;¶
}¶
¶
public Response enterAdminMode(long code) {¶
if (balance != 0) return Response.CANNOT_PERFORM;¶
if (code != id) return Response.INVALID_PARAM;¶
¶
mode = Mode.ADMINISTERING;¶
return Response.OK;¶
}¶
¶
public void exitAdminMode() {¶
mode = Mode.OPERATION;¶
}¶
¶
public Response setPrices(int p1, int p2) {¶
if (mode == Mode.OPERATION) return Response.ILLEGAL_OPERATION;¶
if (p1 <= 0 || p2 <= 0) return Response.INVALID_PARAM;¶
price1 = p1;¶
price2 = p2;¶
return Response.OK;¶
}¶
¶
public Response putCoin1() {¶
if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;¶
if (coins1 == maxc1) return Response.CANNOT_PERFORM;¶
¶
balance += coinval1;¶
```

```
coins1++;¶
¶
return Response.OK;¶
}¶
¶
public Response putCoin2() {¶
if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;¶
if (coins2 == maxc2) return Response.CANNOT_PERFORM;¶
¶
balance += coinval2;¶
coins2++;¶
¶
return Response.OK;¶
}¶
¶
public Response returnMoney() {¶
if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;¶
¶
if (balance == 0) {¶
return Response.OK;¶
} else if (balance > coins1 * coinval1 + coins2 * coinval2) {¶
return Response.TOO_BIG_CHANGE;¶
} else if (balance > coins2 * coinval2) {¶
// using coinval1 == 1¶
coins1 -= (balance - coins2 * coinval2);¶
coins2 = 0;¶
balance = 0;¶
¶
return Response.OK;¶
} else if (balance % coinval2 == 0) {¶
coins2 -= (balance / coinval2);¶
balance = 0;¶
¶
return Response.OK;¶
} else if (coins1 == 0) {¶
// using coinval1 == 1¶
return Response.UNSUITABLE_CHANGE;¶
} else {¶
// using coinval1 == 1¶
coins2 -= (balance / coinval2);¶
coins1--;¶
balance = 0;¶
¶
return Response.OK;¶
}¶
}¶
```

```
¶
public Response giveProduct1(int number) {¶
if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;¶
¶
if (number <= 0 || number > max1) return Response.INVALID_PARAM;¶
if (number > num1) return Response.INSUFFICIENT_PRODUCT;¶
¶
int res = balance - number * price1;¶
¶
if (res < 0) return Response.INSUFFICIENT_MONEY;¶
else if (res > coins1 * coinval1 + coins2 * coinval2) {¶
return Response.TOO_BIG_CHANGE;¶
} else if (res > coins2 * coinval2) {¶
// using coinval1 == 1¶
coins1 -= (res - coins2 * coinval2);¶
coins2 = 0;¶
balance = 0;¶
num1 -= number;¶
¶
return Response.OK;¶
} else if (res % coinval2 == 0) {¶
coins2 -= (res / coinval2);¶
balance = 0;¶
num1 -= number;¶
¶
return Response.OK;¶
} else if (coins1 == 0) {¶
// using coinval1 == 1¶
return Response.UNSUITABLE_CHANGE;¶
} else {¶
// using coinval1 == 1¶
coins2 -= (res / coinval2);¶
coins1--;¶
balance = 0;¶
num1 -= number;¶
¶
return Response.OK;¶
}¶
}¶
¶
public Response giveProduct2(int number) {¶
if (mode == Mode.ADMINISTERING) return Response.ILLEGAL_OPERATION;¶
¶
if (number <= 0 || number > max2) return Response.INVALID_PARAM;¶
if (number > num2) return Response.INSUFFICIENT_PRODUCT;¶
¶
int res = balance - number * price2;¶
¶
if (res < 0) return Response.INSUFFICIENT_MONEY;¶
else if (res > coins1 * coinval1 + coins2 * coinval2) {¶
return Response.TOO_BIG_CHANGE;¶
} else if (res > coins2 * coinval2) {¶
// using coinval1 == 1¶
coins1 -= (res - coins2 * coinval2);¶
coins2 = 0;¶
balance = 0;¶
num2 -= number;¶
¶
return Response.OK;¶
} else if (res % coinval2 == 0) {¶
coins2 -= (res / coinval2);¶
balance = 0;¶
num2 -= number;¶
¶
return Response.OK;¶
} else if (coins1 == 0) {¶
// using coinval1 == 1¶
return Response.UNSUITABLE_CHANGE;¶
} else {¶
// using coinval1 == 1¶
coins1 -= (res / coinval2);¶
coins2--;¶
balance = 0;¶
num2 -= number;¶
¶
return Response.OK;¶
}¶
}¶
¶
}¶
```