Problem Solving With Java #2

By Kevin 2022/2023

Cashier

- Types the item name to record the item and calculate the total price.
- Types `cancel` followed by the items name to cancel an item.
- The system shall also print the product catalog.

```
Items:
apple - $5 banana - $3 carrot - $12.5 durian - $43: watermelon
Sorry no such item!
apple - $5 banana - $3 carrot - $12.5 durian - $43: apple
Shopping cart:
apple
Total: $5.0
apple - $5 banana - $3 carrot - $12.5 durian - $43: banana
Shopping cart:
apple
banana
Total: $8.0
apple - $5 banana - $3 carrot - $12.5 durian - $43: cancel apple
Shopping cart:
apple
banana
apple - Cancelled
Total: $3.0
```

Cashier

Essential Ingredients

- Scanner
- Loop
- Switch

Data to store

- Shopping Cart
- Total

👩‍🏫💬 Again, forget the loop, do a one-off version

Cashier - a no loop version

```
...
public void runOnce() {
    Scanner scanner = new Scanner(System.in);


}
```

👩‍🏫💬 Place the following in order

1. Determine the input
2. Accept user inputs
3. Print the menu
4. Print the Shopping cart
5. Print the total

Cashier - a no loop version

```
Scanner scanner = new Scanner(System.in);
//Print the menu
System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
//accept user inputs
String input = scanner.next();
//determine the input
...
//Print the Shopping cart
System.out.println("Shopping cart:\n" + shoppingCart);
//Print the total
System.out.printf("Total: $%.1f\n", total);
...
```

👩‍🏫💬 We need two variables `shoppingCart` and `total`. What types are they?

Cashier - a no loop version

```
String shoppingCart = "Shopping cart:\n";
float total = 0;
Scanner scanner = new Scanner(System.in);
System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
String input = scanner.next();
//determine the input
...

System.out.println("Shopping cart:\n" + shoppingCart);
System.out.printf("Total: $%.1f\n", total);
```

👨‍🏫💬 Base on value of the input, we do different things. Use `switch` or `if`?

Cashier - a no loop version

```
String shoppingCart = "Shopping cart:\n";
float total = 0;
Scanner scanner = new Scanner(System.in);
System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
String input = scanner.next();

switch (input) {
    case "apple" : total += 5; break; //don't forget your break
    case "banana": total += 3; break;
    case "carrot": total += 12.5; break;
    case "durian": total += 43; break;
    case "cancel": break; //not sure what to do
    default: System.out.println("Sorry no such item!");
}
shoppingCart += input + '\n';

System.out.print("Shopping cart:\n" + shoppingCart); //trailed by \n already
System.out.printf("Total: $%.1f\n", total);
```

Cashier - adding the loop

```
String shoppingCart = "";
float total = 0;
Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
    String input = scanner.next();
    switch (input) {
        case "apple" : total += 5; break;
        case "banana": total += 3; break;
        case "carrot": total += 12.5; break;
        case "durian": total += 43; break;
        case "cancel": break; //not sure what to do
        default: System.out.println("Sorry no such item!");
                 continue;  //add continue here, skip printing shopping cart.
    }
    shoppingCart += input + '\n';
    System.out.print("Shopping cart:\n" + shoppingCart);
    System.out.printf("Total: $%.1f\n", total);
}
```

```
...
    switch (input) {
        case "apple" : total += 5; break;
        case "banana": total += 3; break;
        case "carrot": total += 12.5; break;
        case "durian": total += 43; break;
        case "cancel":
                input = scanner.next();
                switch (input) {
                    case "apple" : total -= 5; break;
                    case "banana": total -= 3; break;
                    case "carrot": total -= 12.5; break;
                    case "durian": total -= 43; break;
                    default: System.out.println("Sorry no such item!");
                }
                shoppingCart += input + ' cancelled\n';
                break;
        default: System.out.println("Sorry no such item!");
                continue;  //add continue here, skip printing shopping cart.
    }
```

👩‍🏫💬 very clumsy, and not quite correct too!

Shorter version

```
String shoppingCart = "";
float total = 0;
Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
    String input = scanner.next();
    boolean cancel = false;
    int sign = 1;
    if (input.equals("cancel")) {
        cancel = true;
        input = scanner.next();
        sign = -1;
    }
    switch (input) {
        case "apple" : total += sign * 5; break;
        case "banana": total += sign * 3; break;
        case "carrot": total += sign * 12.5; break;
        case "durian": total += sign * 43; break;
        default: System.out.println("Sorry no such item!");
                 continue;  //add continue here, skip printing shopping cart.
    }
    shoppingCart += input + (cancel ? " cancelled\n" : "\n");
    System.out.print("Shopping cart:\n" + shoppingCart);
    System.out.printf("Total: $%.1f\n", total);
}
```

Refactor a little

```
String shoppingCart = "Shopping cart:\n";
float total = 0;
Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.print("apple - $5 banana - $3 carrot - $12.5 durian - $43:");
    String input = scanner.next();
    int sign = 1;
    if (input.equals("cancel")) {
        sign = -1;
        input = scanner.next();
    }
    switch (input) {
        case "apple" : total += sign * 5; break;
        case "banana": total += sign * 3; break;
        case "carrot": total += sign * 12.5; break;
        case "durian": total += sign * 43; break;
        default: System.out.println("Sorry no such item!");
                 continue;  //add continue here, skip printing shopping cart.
    }
    shoppingCart += input + (sign == -1 ? " cancelled\n" : "\n"); // here
    System.out.print(shoppingCart);
    System.out.printf("Total: $%.1f\n", total);
}
```

Printing Shapes

Right Triangle

```
*
**
***
****
*****
```

Hollow Square

```
*****
*   *
*   *
*   *
*****
```

Pyramid

```
    *
   ***
  *****
 *******
*********
```

Alt. Square

```
*o*o*
o*o*o
*o*o*
o*o*o
*o*o*
```

Suppose you are given the variable `size`.

Essential Ingredients

- Double for-loops

Warm up

```
*****
*****
*****
*****
*****
```

```java
int size = scanner.nextInt();
for (int i = 0; i < size; i++) { //row
    for (int j = 0; j < size; j++) //on each row
        System.out.print("*");
    System.out.println();
}
```

✗ A wrong solution

```java
int size = scanner.nextInt();
for (int i = 0, j = 0; i < size && j < size; i++, j++)
    System.out.print("*");
```

iteration	1	2	3	4	5
i	0	1	2	3	4
j	0	1	2	3	4

Right Triangle

How many rows? How many stars to print on the i-th row?

```
for (int i = 0; i < ______; i++) {
    for (int j = 0; j < ______; j++)
        System.out.print("*");
    System.out.println();
}
```

Hollow Square

Except the top and the bottom rows, each row has exactly two asterisks and x's spaces. What is x?

```
*****
*   *
*   *
*   *
*****
```

```
for (int j = 0; j < size; j++) //top row
    System.out.print('*');
System.out.println();
for (int i = 1; i < size - 1; i++) { //exclude the top and the bottom

    //?

}
for (int j = 0; j < size; j++) //bottom row
    System.out.print('*');
```

Pyramid

Each row has a few spaces and asterisks `*`. How many?

```
    *
   ***
  *****
 *******
*********
```

Row/ `i`	Leading Spaces	Asterisk
0	4	1
1	3	3
2	2	5
3	1	7
4	0	9

```java
for (int i = 0; i < size; i++) {
    for (int j = 0; j < _______ ; j++)
        System.out.print(' ');
    for (int j = 0; j < _______ ; j++)
        System.out.print('*');
    System.out.println();
}
```

Alt Square

An easier understandable solution

```
*o*o*
o*o*o
*o*o*
o*o*o
*o*o*
```

```java
for (int i = 0; i < size; i++) {
    if (i % 2 == 0) {
        for (int j = 0; j < size; j++) {
            if (j % 2 == 0)
                System.out.print('*');
            else
                System.out.print('o');
        }
    } else {
        for (int j = 0; j < size; j++) {
            if (j % 2 == 1) //alternative row
                System.out.print('*');
            else
                System.out.print('o');
        }
    }
    System.out.println();
}
```

Alt Square

A shorter solution

```
for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++)
        System.out.print( (_______ ? '*' : 'o') );
    System.out.println();
}
```

👩‍🏫💬 Fill the _____ !

Crossing the Bridge Game

Ref: https://www.inwebson.com/demo/cross-the-bridge/

- 6 family members need to cross a bridge within 30 seconds.
- Need to bring a lamp with them (someone need to take the lamp back)
- Times required to cross the bridge for different members are different:
- Max two people can cross a bridge at the same time
- These two people will walk at the same pace.

People	Time Require To Cross Bridge
Alex	1 sec
Bob	2 sec
Carol	4 sec
Dave	6 sec
Eva	8 sec
Fred	12 sec

Crossing the Bridge Game

```
Time: 0
ABCDEF (*) _____________
Enter two initials or one followed by -: A B
Time: 2
CDEF _____________ AB (*)
Enter two initials or one followed by -: D E
Invalid selection!
Enter two initials or one followed by -: D A
Invalid selection!
Enter two initials or one followed by -: A -
Time: 3
ACDEF (*) _____________ B
Enter two initials or one followed by -: C F
Time: 15
ADE _____________ BCF (*)
...
```

Crossing the Bridge Game

Strategy

- Keeping the states of each person
- Keeping the state of the lamp
- Keeping the time
- Construct without validation first
- Construct a never ending game first

Crossing the Bridge Game - Print

- We use 7 variables to keep the states of each person and the lamp
- Each state is binary, i.e. either left or right.

```
boolean a,b,c,d,e,f,lamp; //true = right
a = b = c = d = e = f = lamp = false;
int time = 0;
while (true) {
    //print time
    System.out.println("Time: " + time);
    //print bridge
    String left = "", right = "";
    if (a) right += "A"; else left += "A";
    if (b) right += "B"; else left += "B";
    if (c) right += "C"; else left += "C";
    if (d) right += "D"; else left += "D";
    if (e) right += "E"; else left += "E";
    if (f) right += "F"; else left += "F";
    if (lamp) right += " (*)"; else left += " (*)";
    System.out.println(left + " _________ " + right);
    ...
}
```

Crossing the Bridge Game - Cross a bridge

- `a = !a` allow us to turn true-to-false or false-to-true.

```
...
while (true) {
    //print time and bridge
    ...

    //take input
    System.out.print("Enter two initials..");
    String s1 = scanner.next();
    String s2 = scanner.next();
```

```
    //crossing the bridge
    int s1Time = 0, s2Time = 0;
    switch (s1) {
        case "A": a = !a; s1Time = 1; break;
        case "B": b = !b; s1Time = 2; break;
        case "C": c = !c; s1Time = 4; break;
        case "D": d = !d; s1Time = 6; break;
        case "E": e = !e; s1Time = 8; break;
        case "F": f = !f; s1Time = 12; break;
    }
    switch (s2) {
        case "A": a = !a; s2Time = 1; break;
        case "B": b = !b; s2Time = 2; break;
        case "C": c = !c; s2Time = 4; break;
        case "D": d = !d; s2Time = 6; break;
        case "E": e = !e; s2Time = 8; break;
        case "F": f = !f; s2Time = 12; break;
    }
    time += s1Time > s2Time ? s1Time : s2Time;
    lamp = !lamp;
}
```

Ending Condition

- The ending condition is rather straight forward - all variables are true.
- so change `while (true)` to

```
while (!(a && b && c && d && e && f && lamp))
```

Validation

- It is invalid if the initials and the lamp are not at the same side;
- It is invalid if the initials got repeated;
- It is invalid if the both symbols are -;
- It is invalid if the symbol is not one of the correct initials or -;

We can't help if the user enter three initials at the same time, unless we are using another API from `Scanner` class.

Validation

```
...
System.out.print("Enter two initials..");
String s1 = scanner.next();
String s2 = scanner.next();

//validation
boolean valid = true;
if (s1.equals(s2))
    valid = false;
boolean state1 = false, state2 = false;
switch (s1) {
    case "A": state1 = a; break;
    case "B": state1 = b; break;
    case "C": state1 = c; break;
    case "D": state1 = d; break;
    case "E": state1 = e; break;
    case "F": state1 = f; break;
    case "-": state1 = lamp; break; //!
    default: valid = false;
}
```

```
switch (s2) {
    case "A": state2 = a; break;
    case "B": state2 = b; break;
    case "C": state2 = c; break;
    case "D": state2 = d; break;
    case "E": state2 = e; break;
    case "F": state2 = f; break;
    case "-": state2 = lamp; break;
    default: valid = false;
}
if (state1 != state2 || state1 != lamp)
    valid = false;
if (!valid) {
    System.out.println("Invalid selection!");
    continue;
}
//crossing the bridge
...
```

A better solution

- Array is a good tool.
- An even better solution over array - integer and bit-wise operator

`state`	7th	6th	5th	4th	3rd	2nd	1st	Remark
Examples	Lamp	A	B	C	D	E	F	
0b1101000	1	1	0	1	0	0	0	A, C, and Lamp on the right side
0b0101010	0	1	0	1	0	1	0	A, C, and E on the right side
0b1111111	1	1	1	1	1	1	1	Finished

Crossing a bridge

`state`	7th	6th	5th	4th	3rd	2nd	1st	Remark
	Lamp	A	B	C	D	E	F	
0b1101000	1	1	0	1	0	0	0	A, C, and Lamp on the right side
0b0101010	0	1	0	1	0	1	0	A, C, and E on the right side

- To flip a bit we use XOR operator `^`, i.e., `state = state ^ 0b0000010;`, which make E crosses the bridge.
- If `state` is 0b1100**0**0, `state ^ 0b0000010` becomes 0b1100**1**0
- If `state` is 0b0101**1**1, `state ^ 0b0000010` becomes 0b0101**0**1

XOR with a 0 does nothing. 0 ^ 0 = 0; 1 ^ 0 = 1;
XOR with a 1 flips a bit! 0 ^ 1 = 1; 1 ^ 1 = 0;

Validate

- Check if `A` `C` and `Lamp` are on the **right** side (all three bits are 1, other does not care)
- We use bit-wise AND `&` operator
- Recall `&` perform bit-wise operation, produce 1 if both are 1.
- `A`-`C`-`Lamp` ▶ `checker = 0b1101000`

`state`	`checker` (A-C-LAMP)	`state` & `checker`	Remark
0b**1**1**0**1100	0b1101000	0b**1**1**0**1000	All on the right
0b**1**0**0**1101	0b1101000	0b**1**0**0**1000	Not all on the right

Validate

- Check if `A` `C` and `Lamp` are all on the **left** side (all three bits are 0, other does not care)
- We use bit-wise NOT `~` operator with `&` operator.
- Bitwise NOT `~` invert all bit from 0 to 1 and 1 to 0.

`state`	`checker` (A-C-LAMP)	`~state`	`~state` & `checker`	Remark
0b**00**1**0**100	0b1101000	0b**11**0**1**011	0b**11**0**1**000	All on the left
0b**1**00**1**101	0b1101000	0b**01**1**0**010	0b**01**0**0**000	Not all on the left

Validate

- The checker is built based on the selection of the user.
- The checker should always contain the lamp.
- The checker should also include the one or two initials selected by the user

```
int checker = 0b1000000; //lamp is set
s1 = Scanner.next();
if (s1.equals("A"))
    checker = checker | 0b100000;
if (s1.equals("B"))
    checker = checker | 0b10000;
...
```

- Both OR operator `|` and XOR operator `^` set a bit to 1.

Complete Solution

```
Scanner scanner = new Scanner(System.in);
int state = 0;
int time = 0;
while (state  != 0b1111111) {
  //print time
  System.out.println("Time: " + time);
  //print bridge
  String left = "", right = "";
  if ((state & 0b100000) != 0) right += "A"; else left += "A";
  if ((state & 0b10000) != 0) right += "B"; else left += "B";
  if ((state & 0b1000) != 0) right += "C"; else left += "C";
  if ((state & 0b100) != 0) right += "D"; else left += "D";
  if ((state & 0b10) != 0) right += "E"; else left += "E";
  if ((state & 0b1) != 0) right += "F"; else left += "F";
  if ((state & 0b1000000) != 0)
      right += " (*)";
  else
      left += " (*)";
  System.out.println(left + " _________ " + right);
  System.out.print("Enter two initials..");
  String s1 = scanner.next();
  String s2 = scanner.next();
  boolean valid = true;
  if (s1.equals(s2))
      valid = false;
  int checker = 0b1000000; //always with a lamp
  int s1Time = 0, s2Time = 0;
```

```
  switch (s1) {
      case "A": checker |= 0b100000; s1Time = 1; break;
      case "B": checker |= 0b10000; s1Time = 2; break;
      case "C": checker |= 0b1000; s1Time = 4; break;
      case "D": checker |= 0b100; s1Time = 6; break;
      case "E": checker |= 0b10; s1Time = 8; break;
      case "F": checker |= 0b1; s1Time = 12; break;
      case "-": break;
      default: valid = false;
  }
  switch (s2) {
      case "A": checker |= 0b100000; s2Time = 1; break;
      case "B": checker |= 0b10000; s2Time = 2; break;
      case "C": checker |= 0b1000; s2Time = 4; break;
      case "D": checker |= 0b100; s2Time = 6; break;
      case "E": checker |= 0b10; s2Time = 8; break;
      case "F": checker |= 0b1; s2Time = 12; break;
      case "-": break;
      default: valid = false;
  }
  if ( (state & checker) != checker &&
       ((~state) & checker) != checker )
      valid = false;
  if (!valid) {
      System.out.println("Invalid selection!");
      continue;
  }
  time += s1Time > s2Time ? s1Time : s2Time;
  state = state ^ checker;
}
System.out.println("Finish! Total seconds: " + time);
```