Consider an application for home tuitions. A student can enroll for multiple subjects. A tutor can take many subjects. Each tutor has a unique ide
created.

tutor(tid,name,number)
student(stud_name,number,address)
subject(subject_id,subj_name)
allocation(tid,stud_name,subject_id, fees)

Identify the primary key for the allocation table.

- ○ {tid,stud_name}
- ○ {tid,fees}
- ○ {tid,stud_name,subject_id}
- ○ {tid}

Reset Save

You're

```
class Tester {
    public static void main(String args[]) {
        ConcreteClass obj = new ConcreteClass("AJ", 1);
        System.out.println( obj.getNumber() );
        System.out.println( obj.getLabel() );
    }
}
```

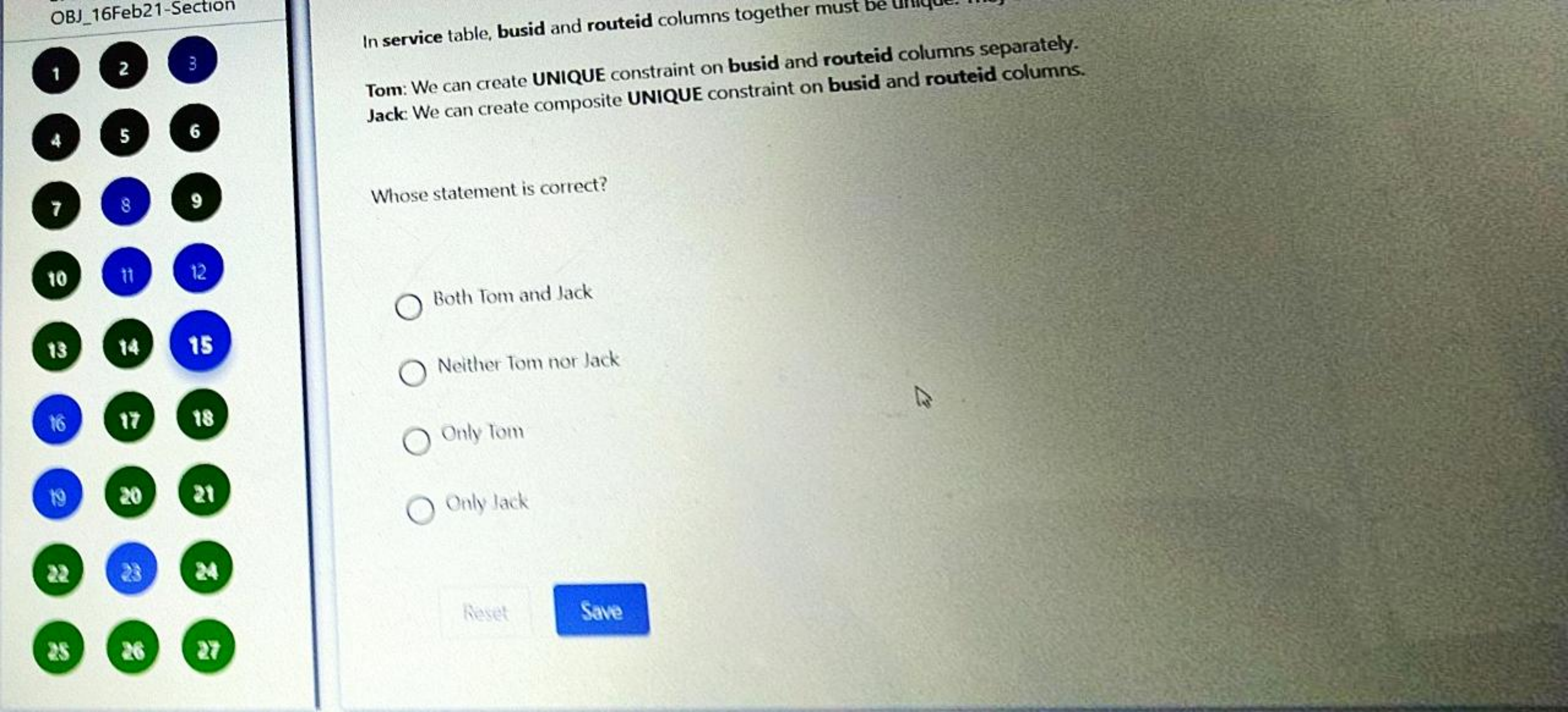
○
```
1
AJ
```

○
```
AJ
1
```

○ Compile Error: getNumber() doesn't have implementation in Abstract class

○ Compile Error: cannot convert from int to String

Reset Save


OBJ_16Feb21-Section

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 26 27

In **service** table, **busid** and **routeid** columns together must be

Tom: We can create **UNIQUE** constraint on **busid** and **routeid** columns separately.
Jack: We can create composite **UNIQUE** constraint on **busid** and **routeid** columns.

Whose statement is correct?

- ○ Both Tom and Jack
- ○ Neither Tom nor Jack
- ○ Only Tom
- ○ Only Jack

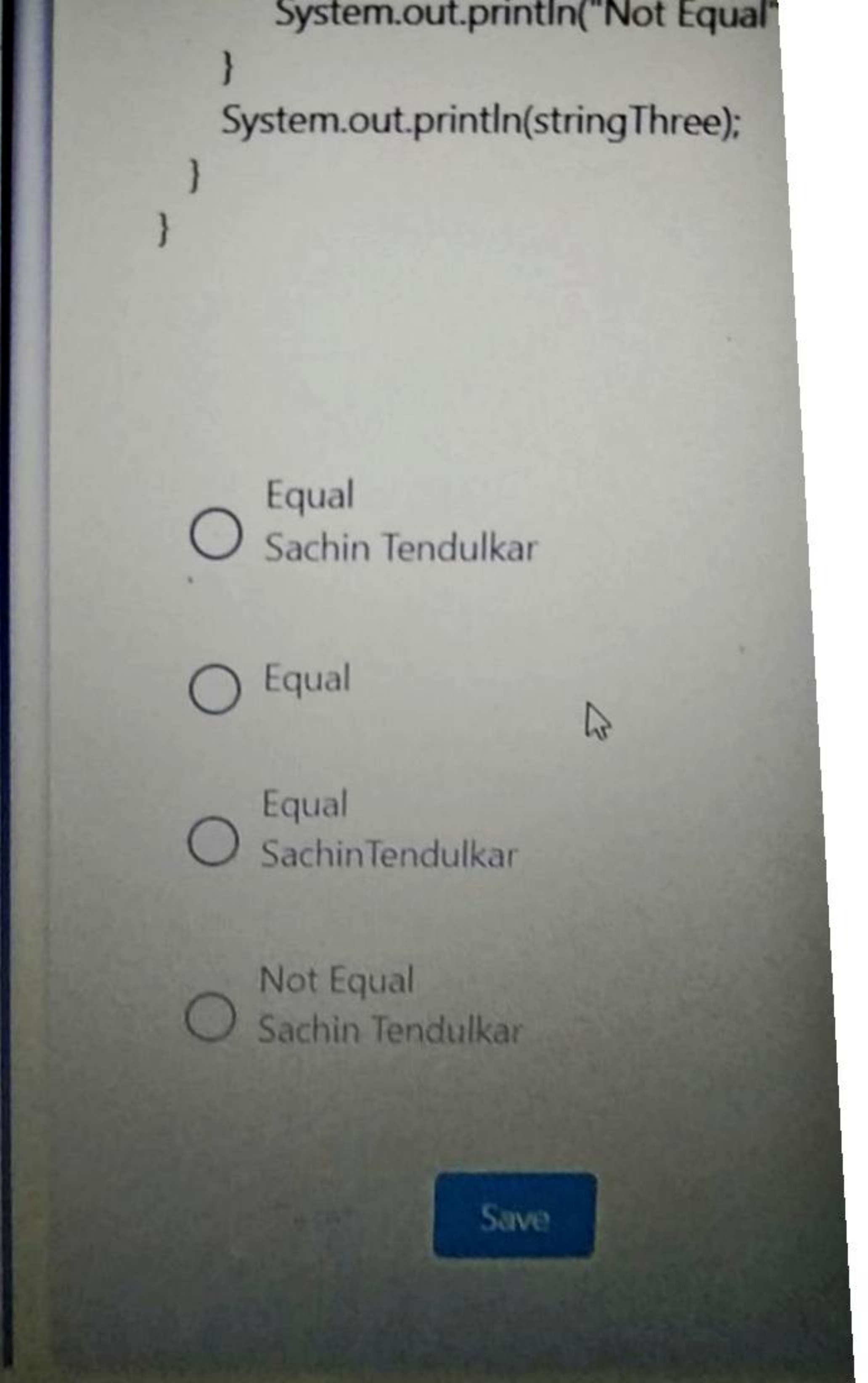
Which of the following 2 equivalent queries retrieves the expected output?

☐
```
SELECT DISTINCT p.productid, o.paymode, SUM(p.price) totalprice
FROM product p RIGHT OUTER JOIN orders o ON p.productid = o.productid
WHERE paymode IS NOT NULL GROUP BY p.productid, o.paymode;
```

☐
```
SELECT DISTINCT p.productid, o.paymode, SUM(p.price) totalprice
FROM product p LEFT OUTER JOIN orders o ON p.productid = o.productid
GROUP BY p.productid, o.paymode;
```

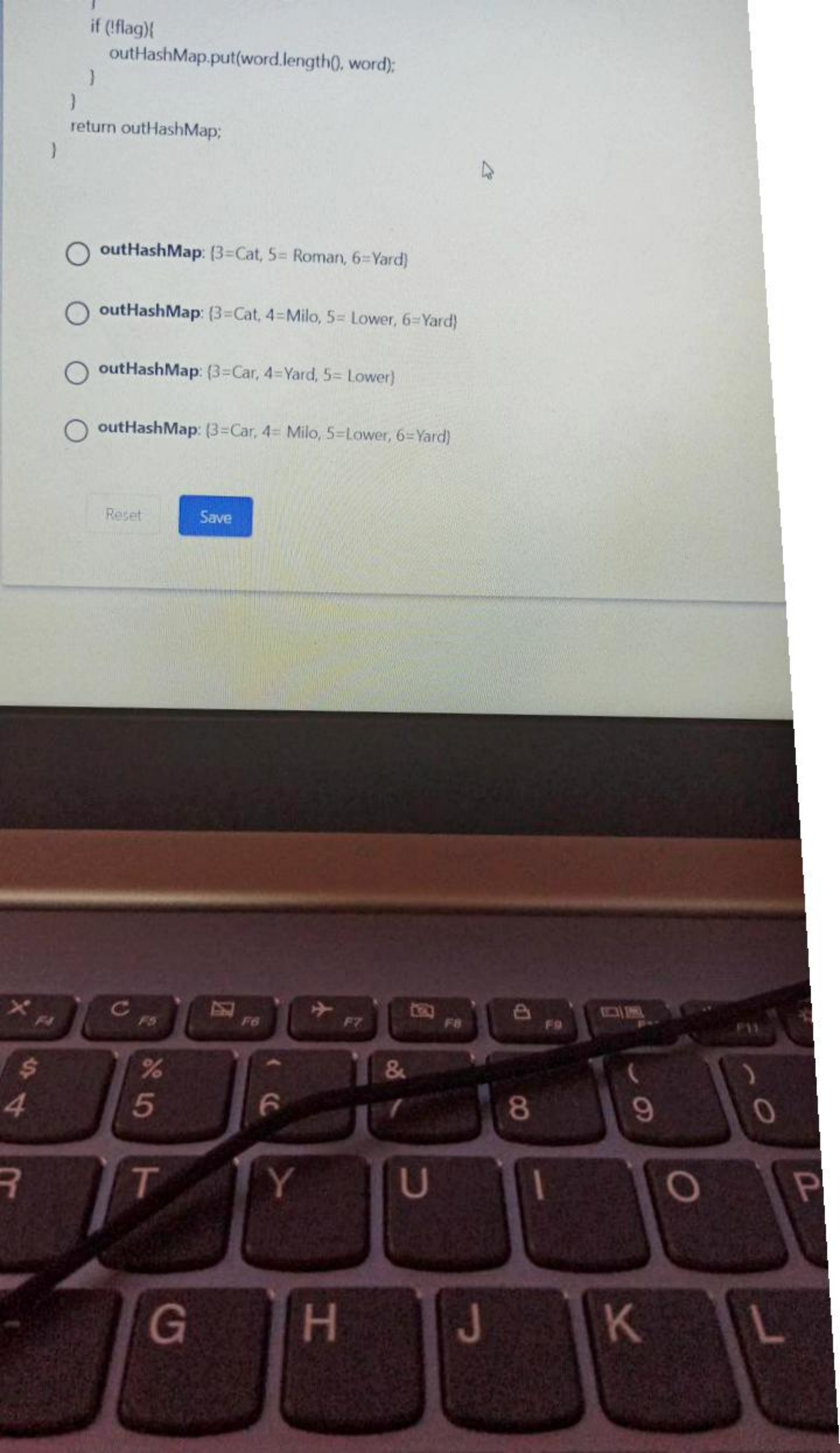
☐
```
SELECT DISTINCT p.productid, o.paymode, SUM(p.price) totalprice
FROM product p INNER JOIN orders o ON p.productid = o.productid
GROUP BY p.productid, o.paymode;
```

☐
```
SELECT DISTINCT p.productid, o.paymode, SUM(p.price) totalprice
FROM orders o RIGHT OUTER JOIN product p ON p.productid = o.productid
AND paymode IS  NOT NULL GROUP BY p.productid, o.paymode;
```



```
            System.out.println("Not Equal"
        }
        System.out.println(stringThree);
    }
}
```

- ○ Equal
  Sachin Tendulkar
- ○ Equal
- ○ Equal
  SachinTendulkar
- ○ Not Equal
  Sachin Tendulkar



```
        if (!flag){
            outHashMap.put(word.length(), word);
        }
    }
    return outHashMap;
}
```

- ○ **outHashMap**: {3=Cat, 5= Roman, 6=Yard}
- ○ **outHashMap**: {3=Cat, 4=Milo, 5= Lower, 6=Yard}
- ○ **outHashMap**: {3=Car, 4=Yard, 5= Lower}
- ○ **outHashMap**: {3=Car, 4= Milo, 5=Lower, 6=Yard}

```sql
SELECT DISTINCT vehiclemodel,bookingid, bookingamount

FROM vehicle v INNER JOIN booking b

ON v.vehicleid = b.vehicleid AND bookingamount > 1000

FULL OUTER JOIN customer c ON b.customerid = c.customerid;
```

How many rows will be fetched when the above query is executed?

- ○ 2
- ○ 3
- ○ 4
- ○ 5

Reset Save

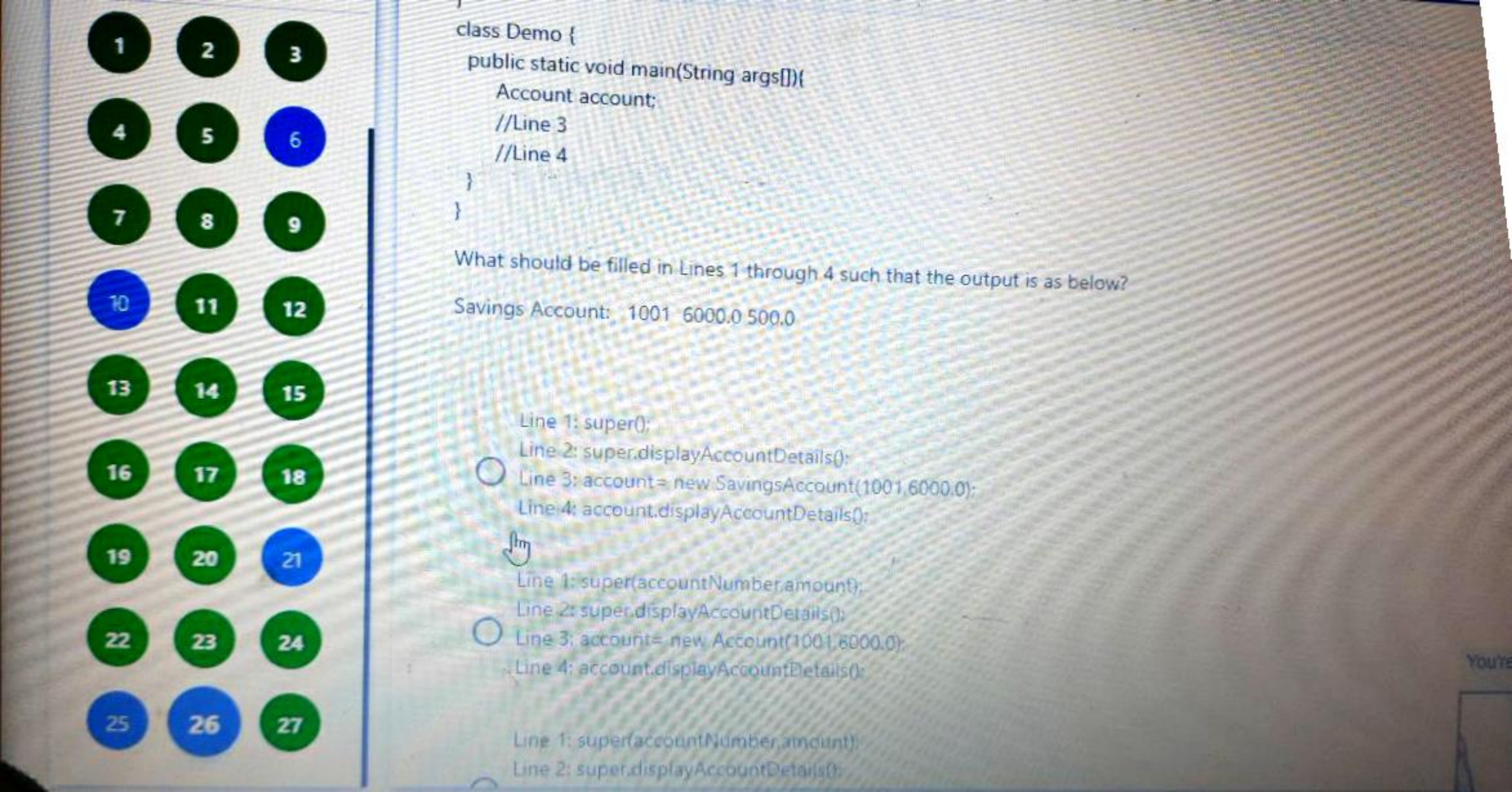
subject(subject_id,subj_name)

allocation(tid,stud_name,subject_id, fe

Identify the primary key for the allocat

- ○ {tid,stud_name}
- ○ {tid,fees}
- ○ {tid,stud_name,subject_id}
- ○ {tid}

Reset Save



```
class Demo {
  public static void main(String args[]){
     Account account;
     //Line 3
     //Line 4
  }
}
```

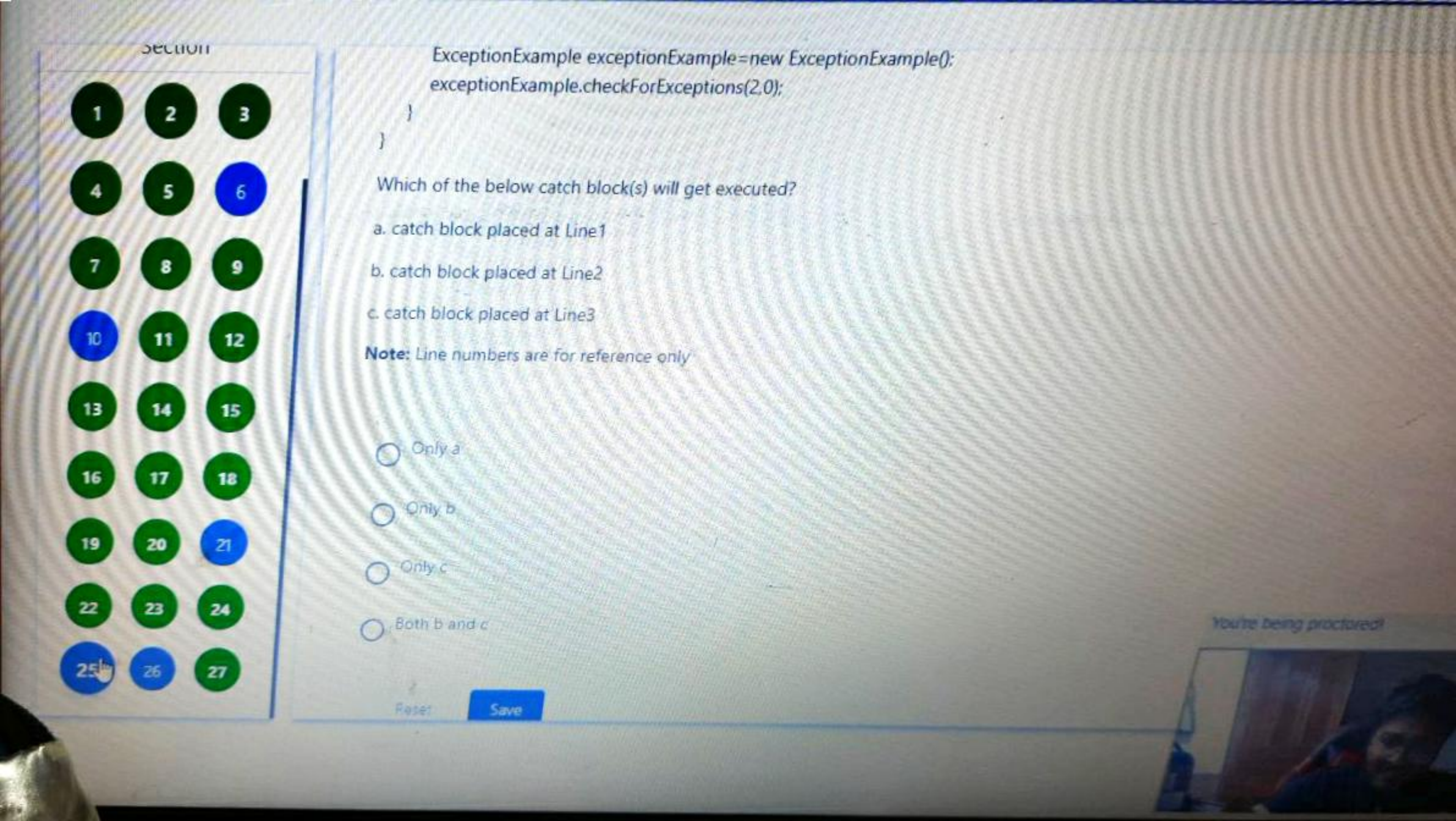
What should be filled in Lines 1 through 4 such that the output is as below?

```
Savings Account:  1001  6000.0 500.0
```

○ Line 1: super();
Line 2: super.displayAccountDetails();
Line 3: account= new SavingsAccount(1001,6000.0);
Line 4: account.displayAccountDetails();

○ Line 1: super(accountNumber,amount);
Line 2: super.displayAccountDetails();
Line 3: account= new Account(1001,6000.0);
Line 4: account.displayAccountDetails();

Line 1: super(accountNumber,amount);
Line 2: super.displayAccountDetails();



```
        ExceptionExample exceptionExample=new ExceptionExample();
        exceptionExample.checkForExceptions(2,0);
    }
}
```

Which of the below catch block(s) will get executed?

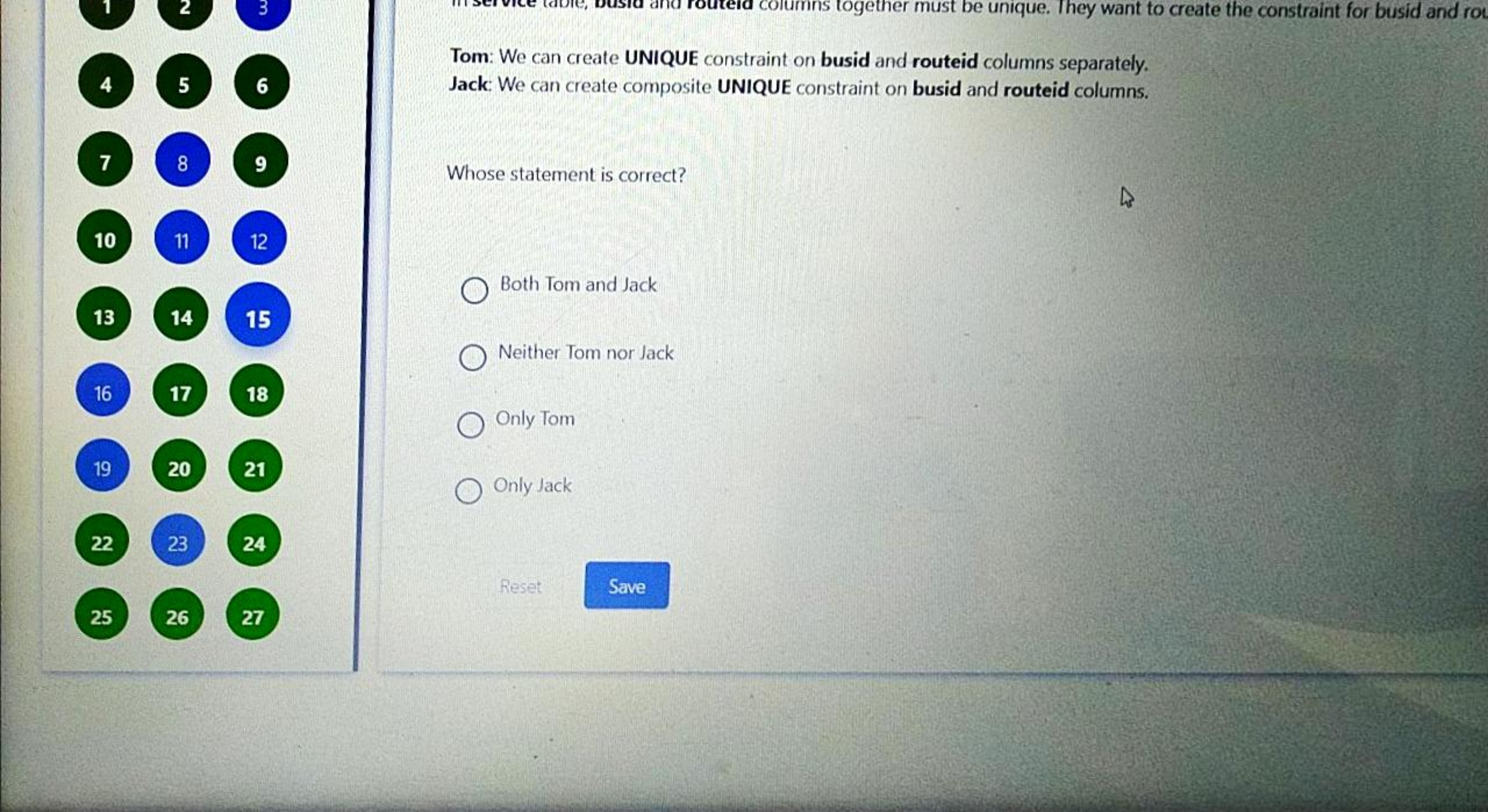
a. catch block placed at Line1

b. catch block placed at Line2

c. catch block placed at Line3

Note: Line numbers are for reference only

- ○ Only a
- ○ Only b
- ○ Only c
- ○ Both b and c



In **service** table, **busid** and **routeid** columns together must be unique. They want to create the constraint for busid and rou

Tom: We can create **UNIQUE** constraint on **busid** and **routeid** columns separately.
Jack: We can create composite **UNIQUE** constraint on **busid** and **routeid** columns.

Whose statement is correct?

○ Both Tom and Jack

○ Neither Tom nor Jack

○ Only Tom

○ Only Jack

Reset Save

```
        System.out.println( obj.getNumber() );
        System.out.println( obj.getLabel() );
    }
}
```

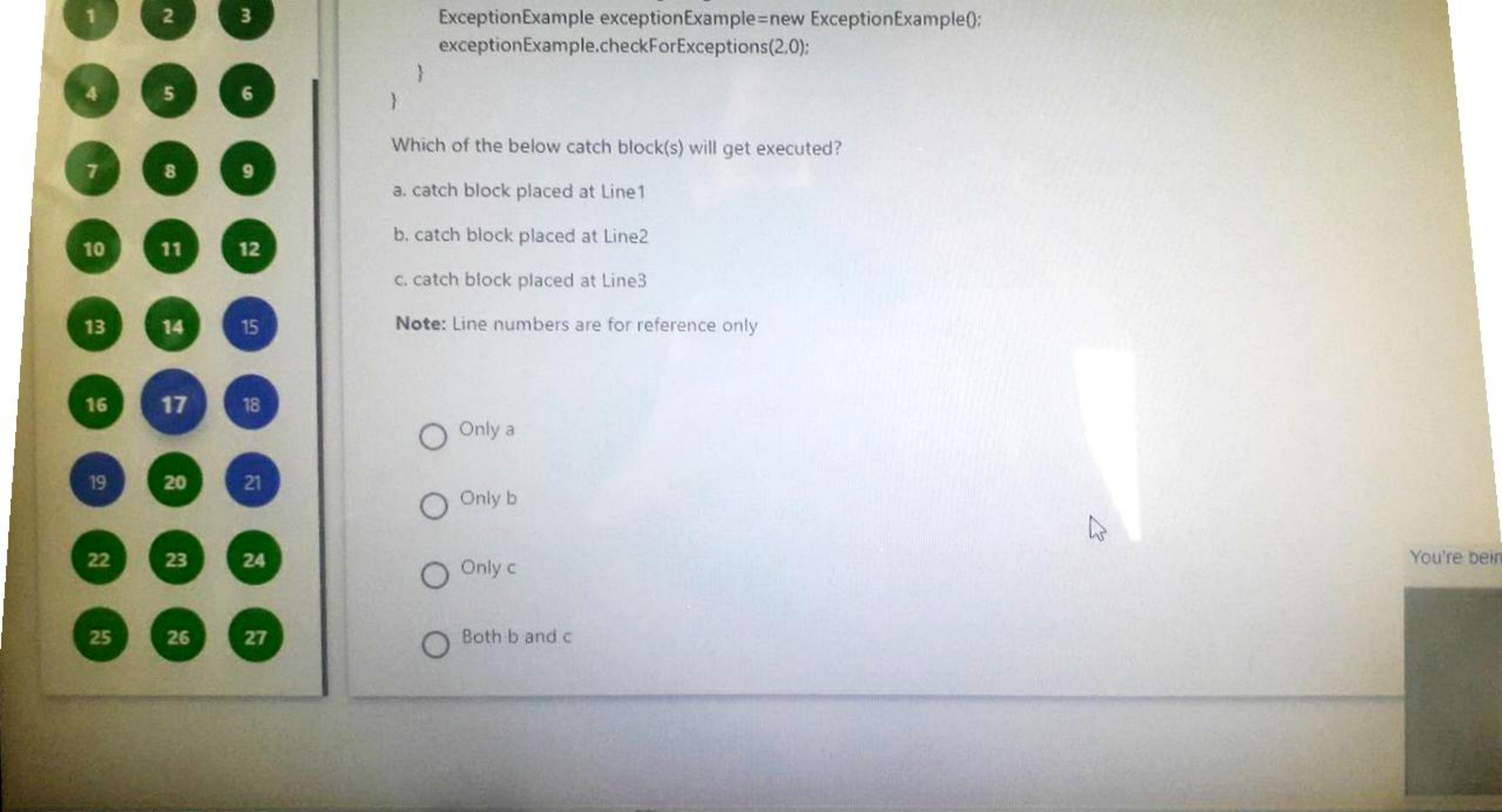
○
```
1
AJ
```

○
```
AJ
1
```

○ Compile Error: getNumber() doesn't have implementation in Abstract class

○ Compile Error: cannot convert from int to String

Reset   Save



```
        ExceptionExample exceptionExample=new ExceptionExample();
        exceptionExample.checkForExceptions(2,0);
    }
}
```

Which of the below catch block(s) will get executed?

a. catch block placed at Line1

b. catch block placed at Line2

c. catch block placed at Line3

Note: Line numbers are for reference only

- ○ Only a
- ○ Only b
- ○ Only c
- ○ Both b and c

table is a foreign key refere

of the following update statements will execute suc

se TWO correct options]

UPDATE customer **SET** customerid = 'C107' **WHERE**

UPDATE dietplan **SET** planid = 'P104' **WHERE** planid='

UPDATE customer **SET** regdate = **sysdate** **WHERE** cus

UPDATE dietplan **SET** customerid = 'C104' **WHERE** custo

Reset

Save

What must be written in Line 1 and Line 2 s

Note: Line numbers are for reference only.

- ○ Line 1: Assert.assertNotEquals(expecte
  Line 2: Assert.assertEquals(expected, ac
- ○ Line 1: Assert.assertEquals(expected, act
  Line 2: Assert.assertNotEquals(expected,
- ○ Line 1: Assert.assertTrue(expected, actual)
  Line 2: Assert.assertFalse(expected, actual)
- ○ Line 1: Assert.assertEquals(expected, actual
  Line 2: Assert.assertFalse(expected, actual

[1 marks]

Consider the line given below which results in compila

```
byte num=200;
```

How can the compilation error be resolved?

Choose THREE CORRECT statements from the below op

- ☐ Change the datatype from byte to int
- ☐ Change the value from 200 to any positive value less t
- ☐ Type cast the value 200 to byte
- ☐ Replace 200 with 200.0

Mark:

```
INSERT INTO service  VALUES(103, NULL, 'No', 1700);
```

Steve:

```
INSERT INTO service  VALUES(101, 703, 'No', 1700);
```

Choose the option that correctly identifies the outcome of

○ Mark 's insert succeeds whereas Steve's insert fails

○ Mark's insert fails due to PRIMARY KEY constraint vi

○ Mark's insert fails due to PRIMARY KEY constraint vio

○ Steve's insert succeeds whereas Mark's insert also suc

Reset Save

```
SELECT companyname FROM company WHERE LENG
```

What will be the output when the above query is executed

○

COMPANYNAME
Arnold Limited
Bud and Co
Heli brothers

○

COMPANYNAME
Cajun Delights
Heli brothers

○

Heli brothers

Bud and Co

Arnold Limited

○

COMPANYNAME

Heli brothers

Bud and Co

Arnold Limited

○

COMPANYNAME

Arnold Limited

Bud and Co