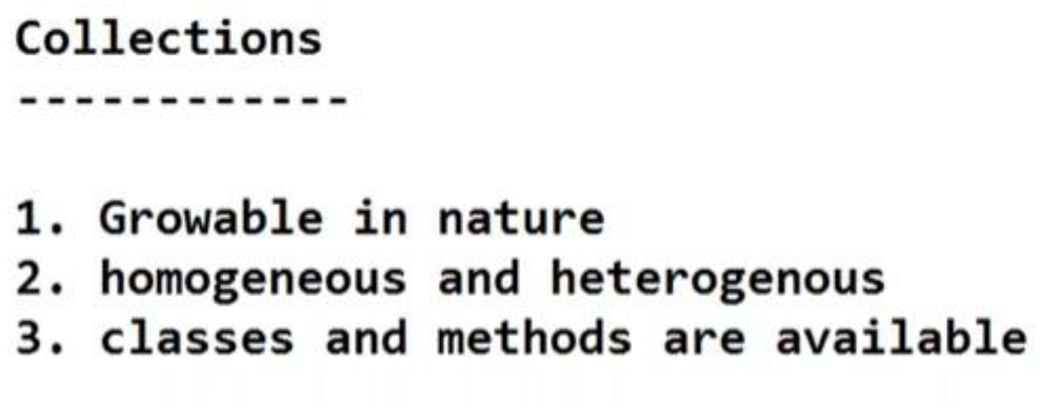
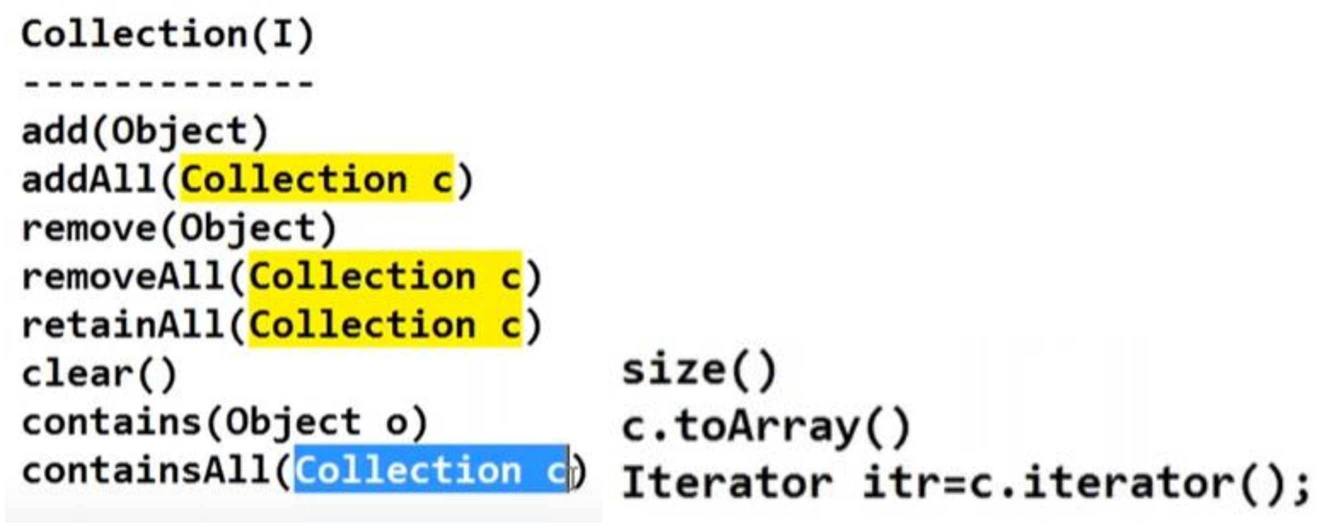
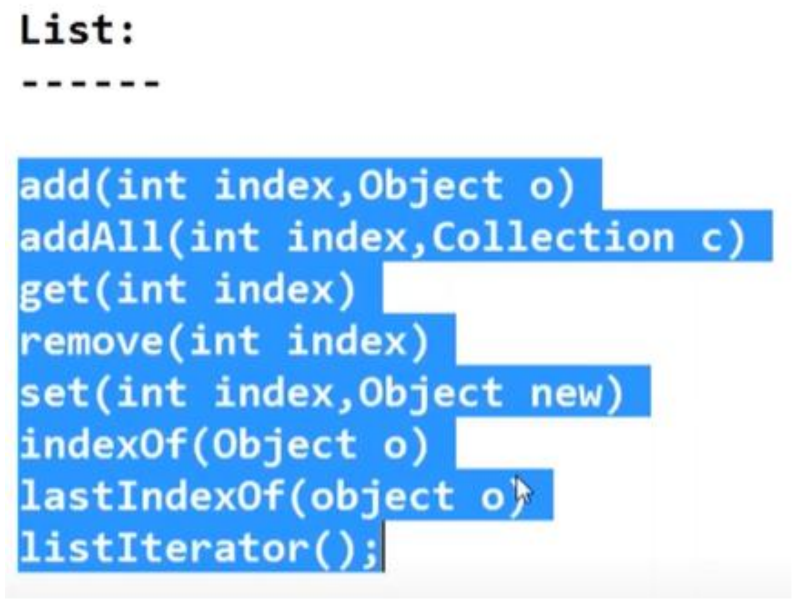
ArrayList



```
Collections
------------

1. Growable in nature
2. homogeneous and heterogenous
3. classes and methods are available
```

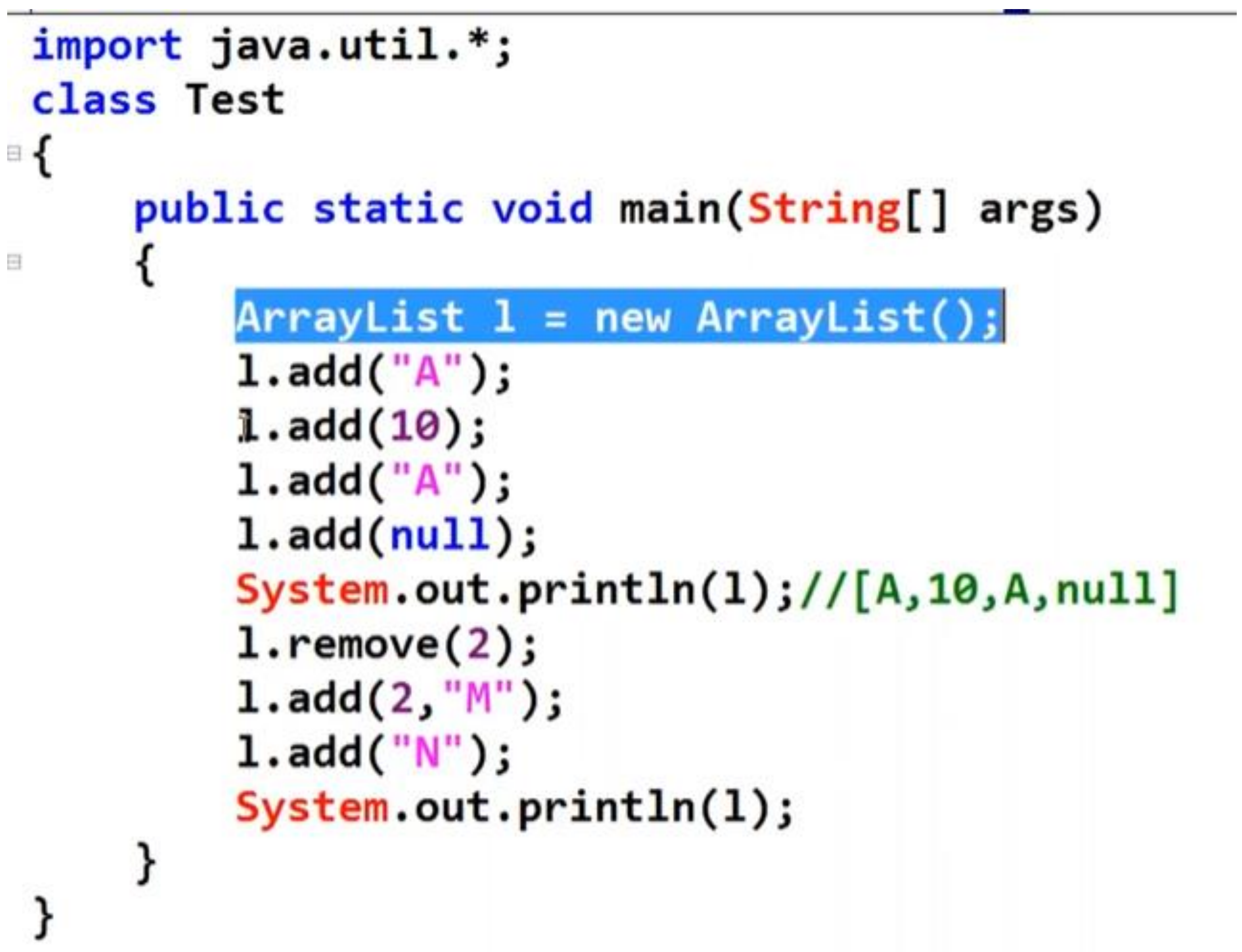


```
Collection(I)
-------------
add(Object)
addAll(Collection c)
remove(Object)
removeAll(Collection c)
retainAll(Collection c)
clear()                    size()
contains(Object o)         c.toArray()
containsAll(Collection c)  Iterator itr=c.iterator();
```



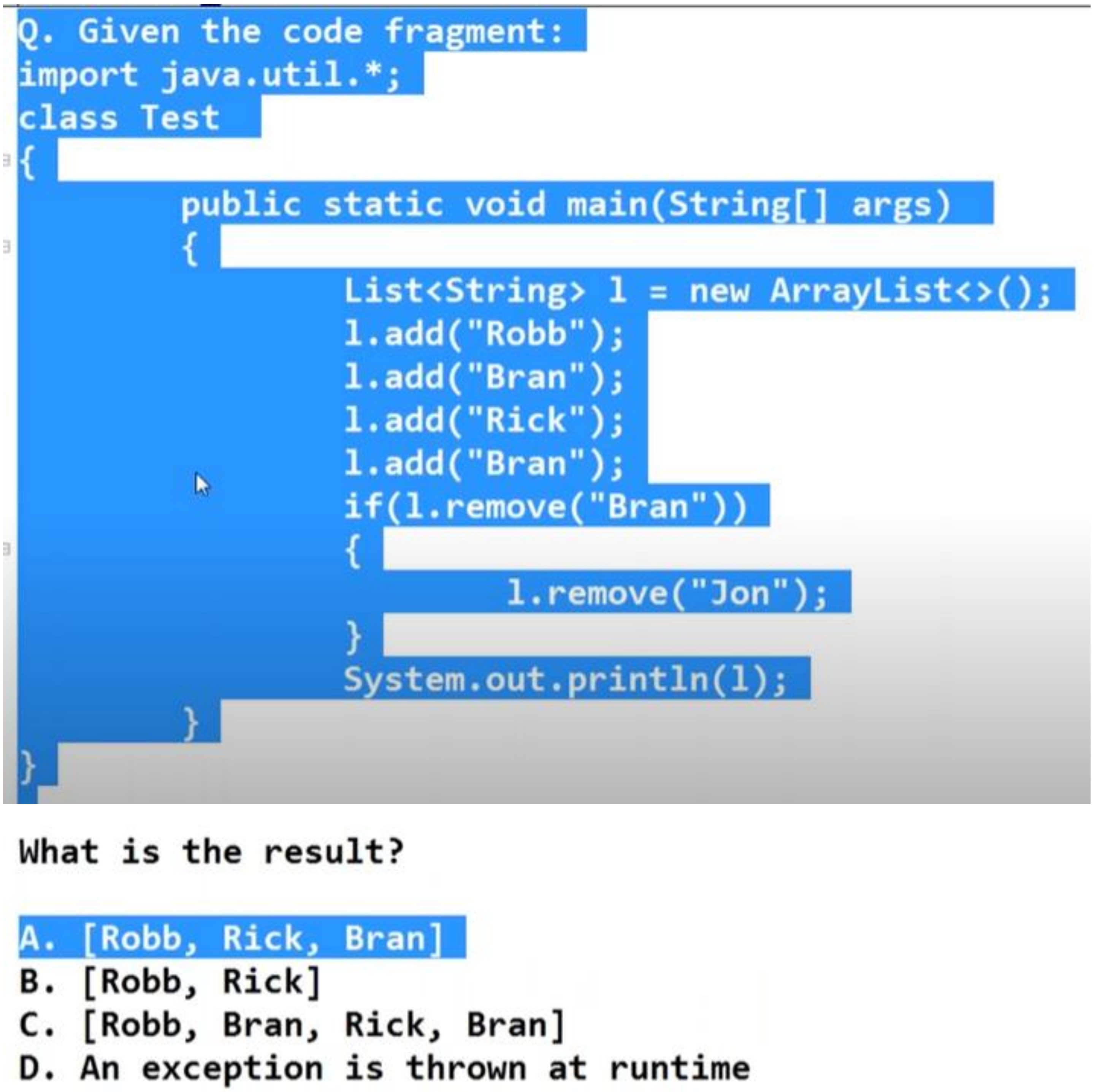
```
List:
------

add(int index,Object o)
addAll(int index,Collection c)
get(int index)
remove(int index)
set(int index,Object new)
indexOf(Object o)
lastIndexOf(object o)
listIterator();
```

```
ArrayList:
----------
Resizable Array or Growable Array
Duplicate objects
Insertion order
null insertion
Heterogeneous
```



```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList l = new ArrayList();
        l.add("A");
        l.add(10);
        l.add("A");
        l.add(null);
        System.out.println(l);//[A,10,A,null]
        l.remove(2);
        l.add(2,"M");
        l.add("N");
        System.out.println(l);
    }
}
```

Q U E S T I O N

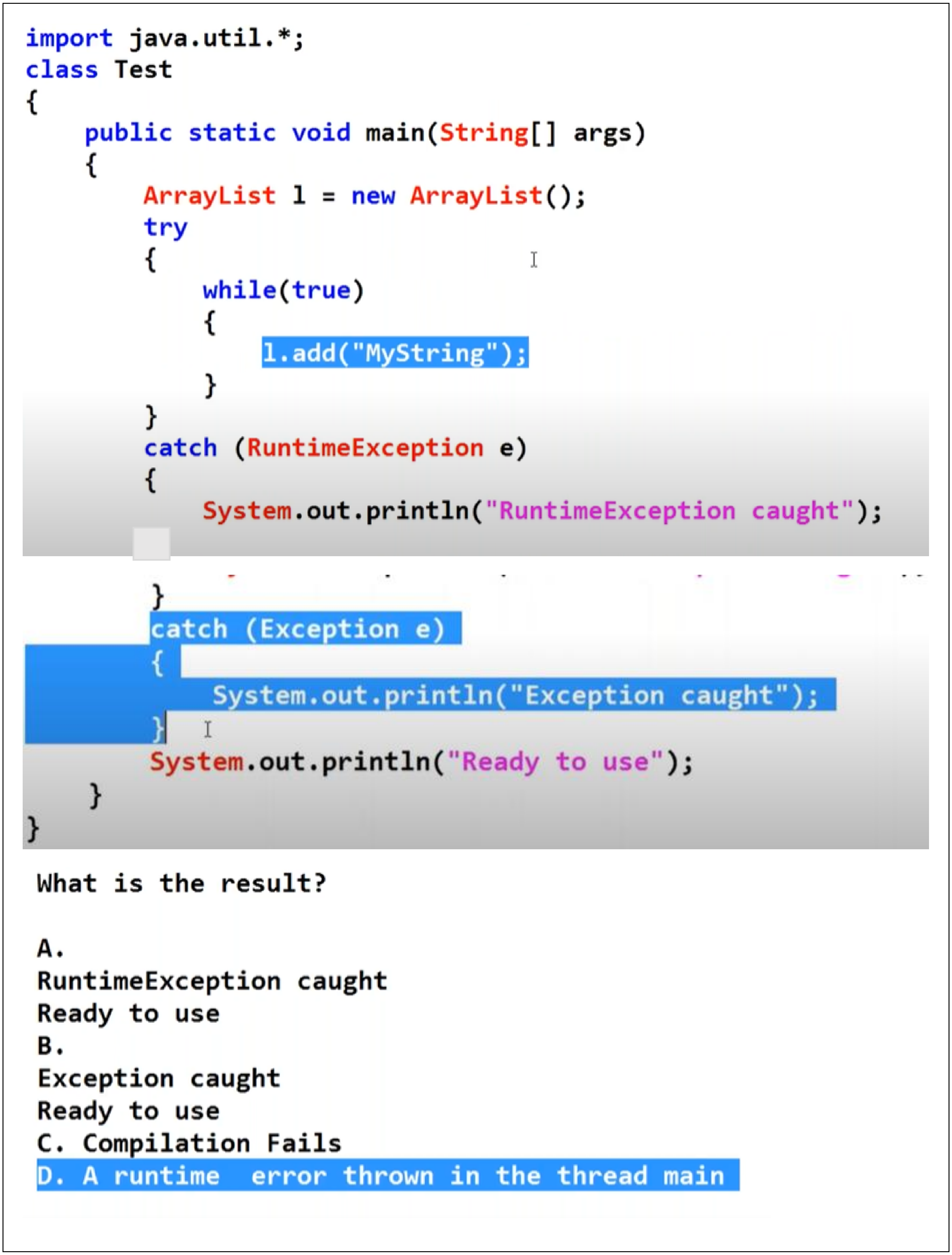


Q. Given the code fragment:

```
import java.util.*;
class Test
{
        public static void main(String[] args)
        {
                List<String> l = new ArrayList<>();
                l.add("Robb");
                l.add("Bran");
                l.add("Rick");
                l.add("Bran");
                if(l.remove("Bran"))
                {
                        l.remove("Jon");
                }
                System.out.println(l);
        }
}
```

What is the result?

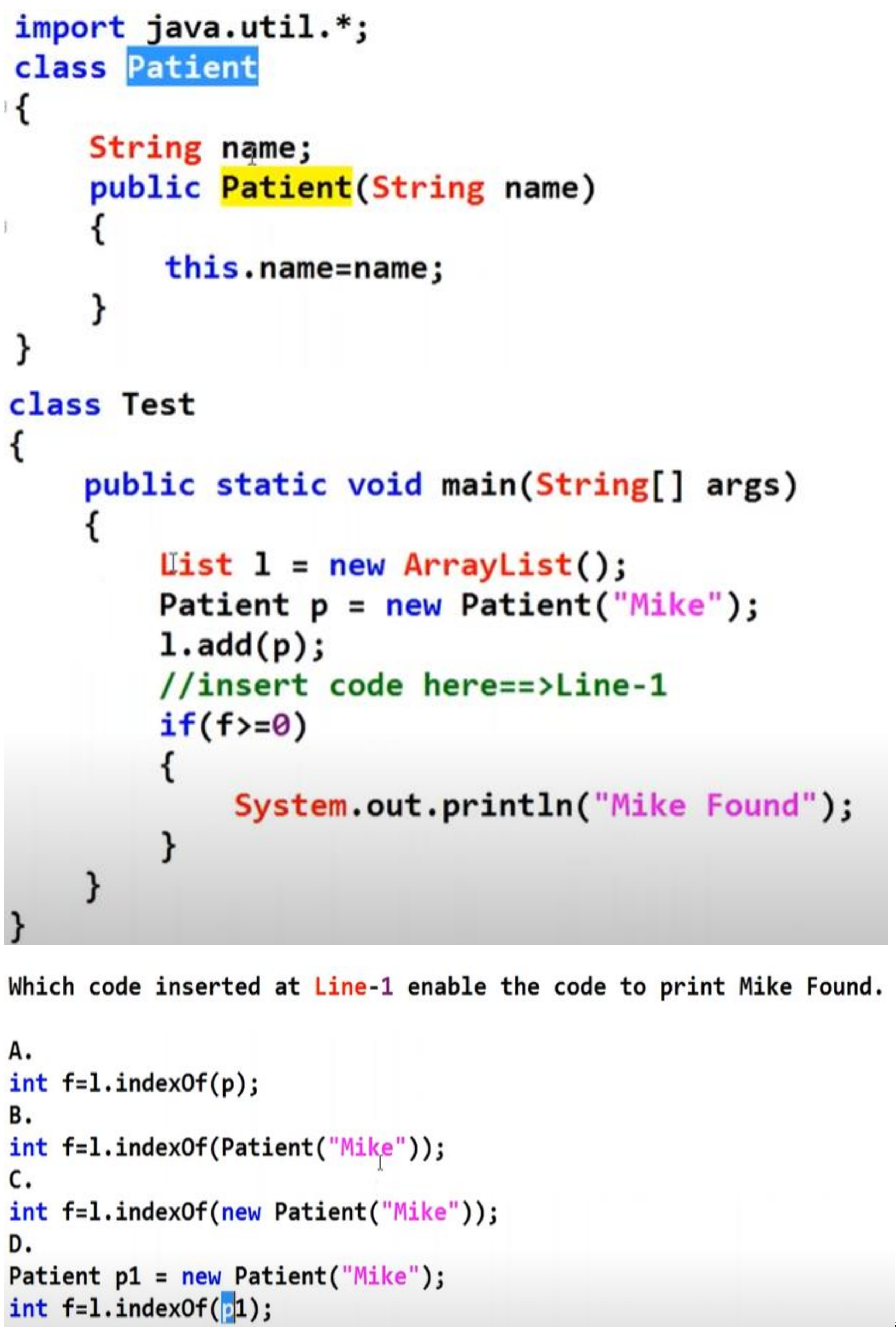
A. [Robb, Rick, Bran]
B. [Robb, Rick]
C. [Robb, Bran, Rick, Bran]
D. An exception is thrown at runtime



```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList l = new ArrayList();
        try
        {
            while(true)
            {
                l.add("MyString");
            }
        }
        catch (RuntimeException e)
        {
            System.out.println("RuntimeException caught");
        }
        catch (Exception e)
        {
            System.out.println("Exception caught");
        }
        System.out.println("Ready to use");
    }
}
```

What is the result?

A.
RuntimeException caught
Ready to use
B.
Exception caught
Ready to use
C. Compilation Fails
D. A runtime error thrown in the thread main

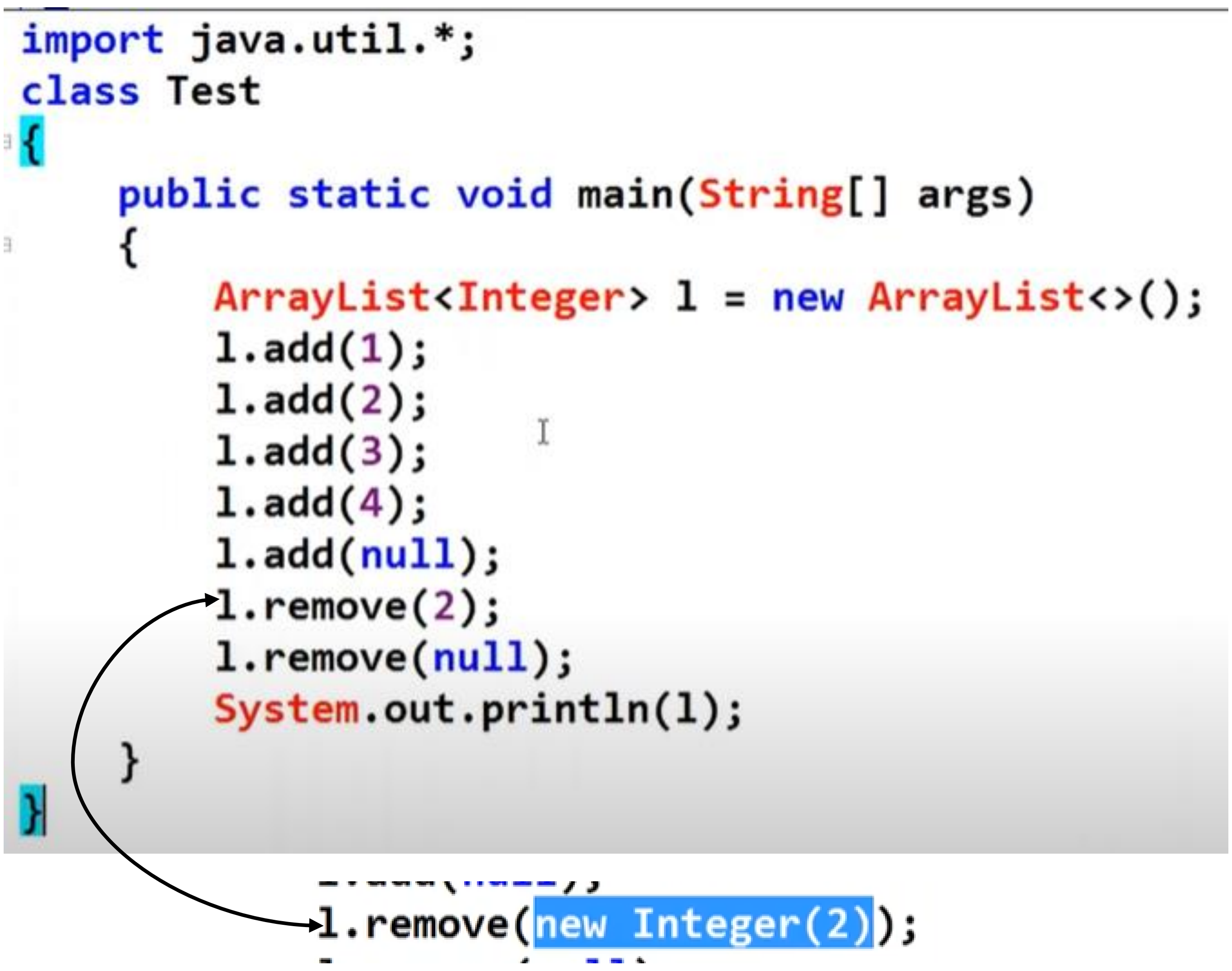


```
import java.util.*;
class Patient
{
    String name;
    public Patient(String name)
    {
        this.name=name;
    }
}
class Test
{
    public static void main(String[] args)
    {
        List l = new ArrayList();
        Patient p = new Patient("Mike");
        l.add(p);
        //insert code here==>Line-1
        if(f>=0)
        {
            System.out.println("Mike Found");
        }
    }
}
```

Which code inserted at Line-1 enable the code to print Mike Found.

A.
```
int f=l.indexOf(p);
```
B.
```
int f=l.indexOf(Patient("Mike"));
```
C.
```
int f=l.indexOf(new Patient("Mike"));
```
D.
```
Patient p1 = new Patient("Mike");
int f=l.indexOf(p1);
```

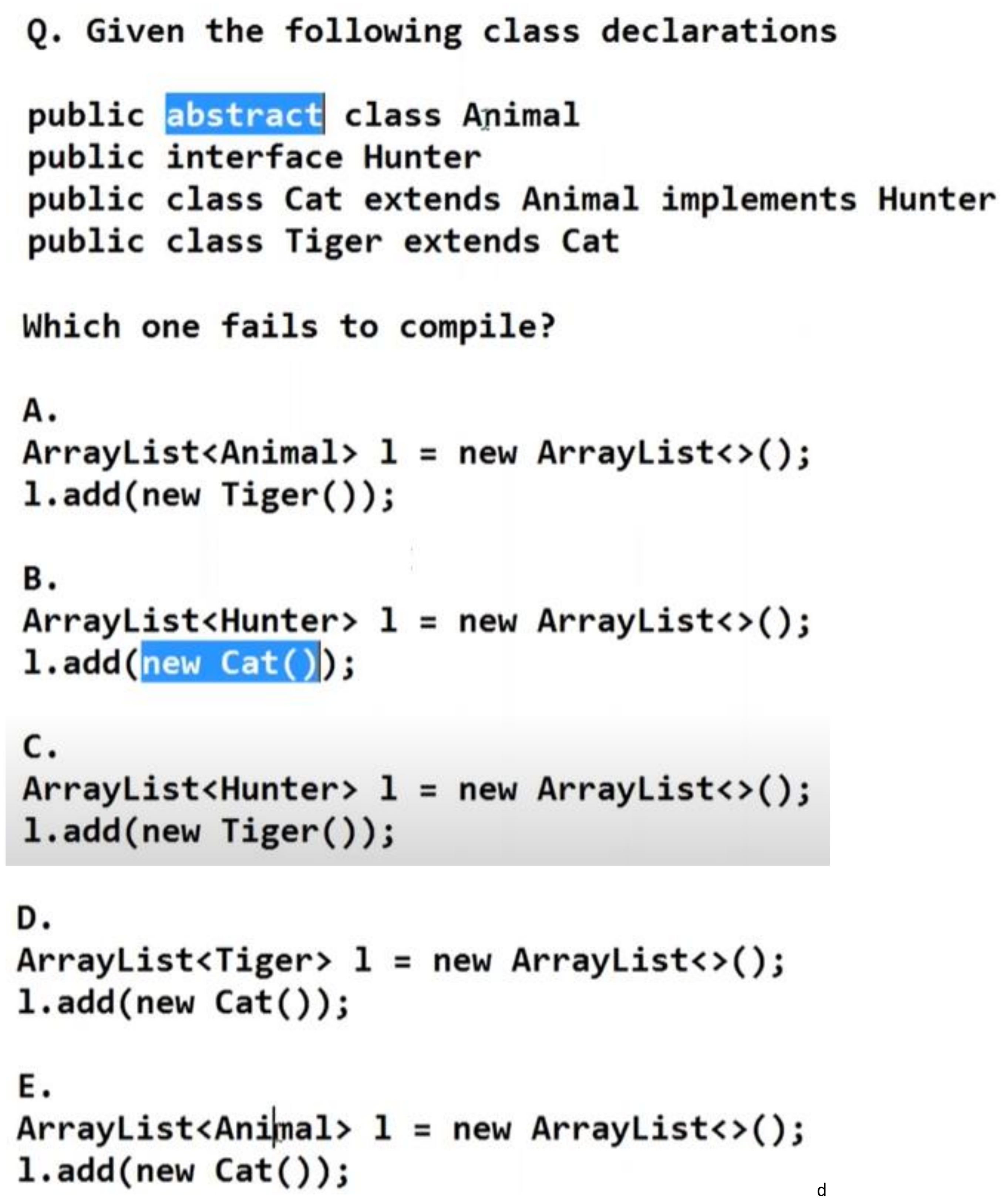
a



```java
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        ArrayList<Integer> l = new ArrayList<>();
        l.add(1);
        l.add(2);
        l.add(3);
        l.add(4);
        l.add(null);
        l.remove(2);
        l.remove(null);
        System.out.println(l);
    }
}
```

```java
l.remove(new Integer(2));
```

```
What is the result?

A. [1, 2, 4]
B. NullPointerException is thrown at runtime
C. [1, 2, 4,null]
D. [1, 3, 4,null]
E. [1, 3, 4]
F. Compilation Fails
```



Q. Given the following class declarations

```
public abstract class Animal
public interface Hunter
public class Cat extends Animal implements Hunter
public class Tiger extends Cat
```

Which one fails to compile?

A.
```
ArrayList<Animal> l = new ArrayList<>();
l.add(new Tiger());
```

B.
```
ArrayList<Hunter> l = new ArrayList<>();
l.add(new Cat());
```

C.
```
ArrayList<Hunter> l = new ArrayList<>();
l.add(new Tiger());
```

D.
```
ArrayList<Tiger> l = new ArrayList<>();
l.add(new Cat());
```

E.
```
ArrayList<Animal> l = new ArrayList<>();
l.add(new Cat());
```

d