

Revision

CS102A Lecture 16

James YU

Dec. 24, 2019



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Question 1

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
2     extends A1 {  
3     protected int a;  
4     private int b;  
5     ...  
6 }
```

```
1 public class A3  
2     extends A2 {  
3     private int q;  
4     ...  
5 }
```

- Which of the following lists of instance data are accessible in class **A2**?

- ① x, y, z, a, b
- ② x, y, z, a
- ③ x, z, a, b
- ④ z, a, b
- ⑤ a, b



Question 1

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
2     extends A1 {  
3     protected int a;  
4     private int b;  
5     ...  
6 }
```

```
1 public class A3  
2     extends A2 {  
3     private int q;  
4     ...  
5 }
```

- Which of the following lists of instance data are accessible in class **A2**?

- ① x, y, z, a, b
- ② x, y, z, a
- ③ x, z, a, b
- ④ z, a, b
- ⑤ a, b

- x, z, a, b

Question 2

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
2     extends A1 {  
3     protected int a;  
4     private int b;  
5     ...  
6 }
```

```
1 public class A3  
2     extends A2 {  
3     private int q;  
4     ...  
5 }
```

- Which of the following lists of instance data are accessible in A3?

- ① x, y, z, a, b, q
- ② a, b, q
- ③ a, q
- ④ x, z, a, q
- ⑤ x, a, q



Question 2

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
2     extends A1 {  
3     protected int a;  
4     private int b;  
5     ...  
6 }
```

```
1 public class A3  
2     extends A2 {  
3     private int q;  
4     ...  
5 }
```

- Which of the following lists of instance data are accessible in A3?

- ① x, y, z, a, b, q
- ② a, b, q
- ③ a, q
- ④ x, z, a, q
- ⑤ x, a, q

- x, z, a, q

Question 3

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
    extends A1 {  
2     protected int a;  
3     private int b;  
4     ...  
5 }
```

```
1 public class A3  
    extends A2 {  
2     private int q;  
3     ...  
4 }
```

- Which of the following is true with the use of instance data **y** of class **A1**?
 - ① it is accessible in **A1**, **A2** and **A3**
 - ② it is accessible in **A1** and **A2**
 - ③ it is accessible only in **A1**
 - ④ it is accessible only in **A3**
 - ⑤ it is not accessible to any of the three classes



Question 3

```
1 public class A1 {  
2     public int x;  
3     private int y;  
4     protected int z;  
5     ...  
6 }
```

```
1 public class A2  
    extends A1 {  
2     protected int a;  
3     private int b;  
4     ...  
5 }
```

```
1 public class A3  
    extends A2 {  
2     private int q;  
3     ...  
4 }
```

- Which of the following is true with the use of instance data **y** of class **A1**?
 - ① it is accessible in **A1**, **A2** and **A3**
 - ② it is accessible in **A1** and **A2**
 - ③ it is accessible only in **A1**
 - ④ it is accessible only in **A3**
 - ⑤ it is not accessible to any of the three classes
- it is accessible only in **A1**



Question 4

- The instruction `super()`; does which of the following?
 - ① calls the method `super` as defined in the current class
 - ② calls the method `super` as defined in the current class' parent class
 - ③ calls the method `super` as defined in `java.lang`
 - ④ calls the constructor as defined in the current class
 - ⑤ calls the constructor as defined in the current class' parent class

Question 4

- The instruction `super();` does which of the following?
 - ① calls the method `super` as defined in the current class
 - ② calls the method `super` as defined in the current class' parent class
 - ③ calls the method `super` as defined in `java.lang`
 - ④ calls the constructor as defined in the current class
 - ⑤ calls the constructor as defined in the current class' parent class
- calls the constructor as defined in the current class' parent class



Question 5

- Aside from permitting inheritance, the visibility modifier `protected` is also used to
 - ① permit access to the protected item by any class defined in the same package
 - ② permit access to the protected item by any `static` class
 - ③ permit access to the protected item by any parent class
 - ④ ensure that the class can not throw a `NullPointerException`
 - ⑤ define abstract elements of an `interface`



Question 5

- Aside from permitting inheritance, the visibility modifier `protected` is also used to
 - ① permit access to the protected item by any class defined in the same package
 - ② permit access to the protected item by any `static` class
 - ③ permit access to the protected item by any parent class
 - ④ ensure that the class can not throw a `NullPointerException`
 - ⑤ define abstract elements of an `interface`
- permit access to the protected item by any class defined in the same package



Question 6

- All classes in Java are directly or indirectly subclasses of the () class.
 - ① Wrapper
 - ② String
 - ③ Reference
 - ④ this
 - ⑤ Object



Question 6

- All classes in Java are directly or indirectly subclasses of the () class.
 - ① Wrapper
 - ② String
 - ③ Reference
 - ④ this
 - ⑤ Object
- Object

Question 7

- The reference to `getMoney()` in assignment 1 is to the class
 - 1 `Person`
 - 2 `Student`
 - 3 `Employee`
 - 4 `Retired`
 - 5 none of the above, this cannot be determined by examining the code

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10     p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```

Question 7

- The reference to `getMoney()` in assignment 1 is to the class
 - ① `Person`
 - ② `Student`
 - ③ `Employee`
 - ④ `Retired`
 - ⑤ none of the above, this cannot be determined by examining the code
- `Person`

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10     p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```

Question 8

- The reference to `getMoney()` in assignment 2 is to the class
 - 1 `Person`
 - 2 `Student`
 - 3 `Employee`
 - 4 `Retired`
 - 5 none of the above, this cannot be determined by examining the code

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10    p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```


Question 8

- The reference to `getMoney()` in assignment 2 is to the class
 - 1 `Person`
 - 2 `Student`
 - 3 `Employee`
 - 4 `Retired`
 - 5 none of the above, this cannot be determined by examining the code
- `Student`

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10     p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```

Question 9

- The reference to `getMoney()` in assignment 3 is to the class
 - 1 `Person`
 - 2 `Student`
 - 3 `Employee`
 - 4 `Retired`
 - 5 none of the above, this cannot be determined by examining the code

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10     p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```



Question 9

- The reference to `getMoney()` in assignment 3 is to the class
 - 1 `Person`
 - 2 `Student`
 - 3 `Employee`
 - 4 `Retired`
 - 5 none of the above, this cannot be determined by examining the code
- none of the above, this cannot be determined by examining the code

```
1 Person p = new Person(...);
2 // assignment 1
3 int m1 = p.getMoney();
4 p = new Student(...);
5 // assignment 2
6 int m2 = p.getMoney();
7 if (m2 < 100000)
8     p = new Employee(...);
9 else if (m1 > 50000)
10     p = new Retired(...);
11 // assignment 3
12 int m3 = p.getMoney();
```



Question 10

- The relationship between a child (sub) class and a parent (super) class is referred to as a(n) () relationship.
 - ① has-a
 - ② is-a
 - ③ was-a
 - ④ instance-of

Question 10

- The relationship between a child (sub) class and a parent (super) class is referred to as a(n) () relationship.
 - ① has-a
 - ② is-a
 - ③ was-a
 - ④ instance-of
- is-a



Question 11

```
1 public class Test {  
2     private int x;  
3     public test (int newValue) {  
4         x = newValue;  
5     }  
6 }
```

- Which of the following is true about the class `Test`?
 - ① it has no parent class
 - ② it's parent class is `Object`
 - ③ it's parent class is Java
 - ④ it can not be extended
 - ⑤ it has a default child called `Object`



Question 11

```
1 public class Test {  
2     private int x;  
3     public test (int newValue) {  
4         x = newValue;  
5     }  
6 }
```

- Which of the following is true about the class `Test`?
 - ① it has no parent class
 - ② it's parent class is `Object`
 - ③ it's parent class is Java
 - ④ it can not be extended
 - ⑤ it has a default child called `Object`
- it's parent class is `Object`



Question 12

```
1 public class Test {  
2     private int x;  
3     public test (int newValue) {  
4         x = newValue;  
5     }  
6 }
```

- If `q1` and `q2` are objects of `Test` class, then `q1.equals(q2)`
 - ① is a syntax error since `equals` is not defined in the `Test` class
 - ② is `true` if `q1` and `q2` both store the same value of `x`
 - ③ is `true` if `q1` and `q2` reference the same `Test` object
 - ④ is never `true`
 - ⑤ throws a `NullPointerException`



Question 12

```
1 public class Test {  
2     private int x;  
3     public test (int newValue) {  
4         x = newValue;  
5     }  
6 }
```

- If `q1` and `q2` are objects of `Test` class, then `q1.equals(q2)`
 - ① is a syntax error since `equals` is not defined in the `Test` class
 - ② is `true` if `q1` and `q2` both store the same value of `x`
 - ③ is `true` if `q1` and `q2` reference the same `Test` object
 - ④ is never `true`
 - ⑤ throws a `NullPointerException`
- is never `true`



Question 13

```
1 for (int j = 0; j < list.length; j++)  
2   if (list[j] < temp)  
3     c++;
```

- What does the code do? Assume `list` is an initialized array of `int` values, `temp` is some previously initialized `int` value, and `c` is an `int` initialized to 0.
 - ① It finds the smallest value and stores it in `temp`
 - ② It finds the largest value and stores it in `temp`
 - ③ It counts the number of elements equal to the smallest value in `list`
 - ④ It counts the number of elements in `list` that are less than `temp`
 - ⑤ It sorts the values in `list` to be in ascending order



Question 13

```
1 for (int j = 0; j < list.length; j++)  
2   if (list[j] < temp)  
3     c++;
```

- What does the code do? Assume `list` is an initialized array of `int` values, `temp` is some previously initialized `int` value, and `c` is an `int` initialized to 0.
 - ① It finds the smallest value and stores it in `temp`
 - ② It finds the largest value and stores it in `temp`
 - ③ It counts the number of elements equal to the smallest value in `list`
 - ④ It counts the number of elements in `list` that are less than `temp`
 - ⑤ It sorts the values in `list` to be in ascending order
- It counts the number of elements in `list` that are less than `temp`



Question 14

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i1 would be caught by the **catch** statement labeled
1) e1; 2) e2; 3) e5; 4) e1 or e5; 5) e1, e4 or e5



Question 14

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i1 would be caught by the `catch` statement labeled
1) e1; 2) e2; 3) e5; 4) e1 or e5; 5) e1, e4 or e5
- e1 or e5



Question 15

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i2 would be caught by the **catch** statement labeled
1) e1; 2) e2; 3) e3; 4) e5; 5) e2 or e5



Question 15

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i2 would be caught by the **catch** statement labeled
1) e1; 2) e2; 3) e3; 4) e5; 5) e2 or e5
- e2 or e5

Question 16

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i3 would be caught by the **catch** statement labeled
1) e2; 2) e3; 3) e4; 4) e3 or e4; 5) e2, e3 or e4



Question 16

```
1 try {  
2     BufferedReader f = new BufferedReader(new FileReader(filename)); //i1  
3     int x = Integer.parseInt(f.readLine()); //i2  
4     a[++i] = (double) (1 / x); //i3  
5 }  
6 catch (FileNotFoundException ex) {...} // e1  
7 catch (NumberFormatException ex) {...} // e2  
8 catch (ArithmeticException ex) {...} // e3  
9 catch (ArrayIndexOutOfBoundsException ex) {...} // e4  
10 catch (IOException ex) {...} // e5
```

- An exception raised by the instruction in i3 would be caught by the `catch` statement labeled
1) e2; 2) e3; 3) e4; 4) e3 or e4; 5) e2, e3 or e4
- e3 or e4



Question 17

- Consider the array declaration and instantiation: `int[] arr = new int[5];`
Which of the following is **true** about **arr**?
 - It stores 5 elements with legal indices between 1 and 5
 - It stores 5 elements with legal indices between 0 and 4
 - It stores 4 elements with legal indices between 1 and 4
 - It stores 6 elements with legal indices between 0 and 5
 - It stores 5 elements with legal indices between 0 and 5



Question 17

- Consider the array declaration and instantiation: `int[] arr = new int[5];`
Which of the following is **true** about **arr**?
 - It stores 5 elements with legal indices between 1 and 5
 - It stores 5 elements with legal indices between 0 and 4
 - It stores 4 elements with legal indices between 1 and 4
 - It stores 6 elements with legal indices between 0 and 5
 - It stores 5 elements with legal indices between 0 and 5
- It stores 5 elements with legal indices between 0 and 4



Question 18

- Which of the following messages passed to the string `str` could throw a `StringIndexOutOfBoundsException`?
 - ① `str.length()`
 - ② `str.charAt(2);`
 - ③ `str.replace('a', 'A');`
 - ④ `str.equals(str);`
 - ⑤ any of the above could throw a `StringIndexOutOfBoundsException`



Question 18

- Which of the following messages passed to the string `str` could throw a `StringIndexOutOfBoundsException`?
 - ① `str.length()`
 - ② `str.charAt(2);`
 - ③ `str.replace('a', 'A');`
 - ④ `str.equals(str);`
 - ⑤ any of the above could throw a `StringIndexOutOfBoundsException`
- `str.charAt(2);`



Question 19

- Assume `x` and `y` are `String` variables with `x = "Hello"` and `y = null`. If the operation `y = "Hello";` is performed, then the result of `(x == y)` is
 - `true`
 - `false`
 - `x` and `y` becoming aliases
 - `x` being set to the value `null`
 - a run-time error



Question 19

- Assume `x` and `y` are `String` variables with `x = "Hello"` and `y = null`. If the operation `y = "Hello";` is performed, then the result of `(x == y)` is
 - `true`
 - `false`
 - `x` and `y` becoming aliases
 - `x` being set to the value `null`
 - a run-time error
- `false`



Question 20

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- What is returned by `values[1]`?



Question 20

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- What is returned by `values[1]`?
- 4



Question 21

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- What is returned by `values.length`?



Question 21

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- What is returned by `values.length`?
- 7



Question 22

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- Which of the following loops would adequately add 1 to each element stored in `values`?

- 1 `for (int j=1; j < values.length ; j++) values[j]++;`
- 2 `for (int j=0; j < values.length ; j++) values[j]++;`
- 3 `for (int j=0; j <= values.length; j++) values[j]++;`
- 4 `for (int j=0; j < values.length-1; j++) values[j]++;`
- 5 `for (int j=1; j < values.length-1; j++) values[j]++;`



Question 22

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- Which of the following loops would adequately add 1 to each element stored in `values`?
 - ① `for (int j=1; j < values.length ; j++) values[j]++;`
 - ② `for (int j=0; j < values.length ; j++) values[j]++;`
 - ③ `for (int j=0; j <= values.length; j++) values[j]++;`
 - ④ `for (int j=0; j < values.length-1; j++) values[j]++;`
 - ⑤ `for (int j=1; j < values.length-1; j++) values[j]++;`
- `for (int j=0; j < values.length ; j++) values[j]++;`



Question 23

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- The statement `System.out.println(values[7]);` will
 - ① output 7
 - ② output 18
 - ③ output nothing
 - ④ cause an `ArrayOutOfBoundsException` to be thrown
 - ⑤ cause a syntax error



Question 23

9	4	12	2	6	8	18
---	---	----	---	---	---	----

- The statement `System.out.println(values[7]);` will
 - ① output 7
 - ② output 18
 - ③ output nothing
 - ④ cause an `ArrayOutOfBoundsException` to be thrown
 - ⑤ cause a syntax error
- cause an `ArrayOutOfBoundsException` to be thrown



Question 24

- Consider that you want to extend `Aclass` to `Bclass`. `Bclass` will have a third `int` instance data, `z`. Which of the following would best define `Bclass`' constructor?

```
1 public Bclass (int a, int b,  
2     int c) {  
3     super(a, b);  
4     z = c;  
5 }
```

```
1 public class Aclass {  
2     private int x;  
3     protected int y;  
4  
5     public Aclass (int a, int b) {  
6         x = a;  
7         y = b;  
8     }  
9  
10    public int addEm () {  
11        return x + y;  
12    }  
13  
14    public String toString () {  
15        return "" + x + " " + y;  
16    }  
17 }
```



Question 25

- Which of the following would best redefine the `toString` method for `BClass`?

```
1 public String toString () {  
2     return super.toString() + "  
3     " + z;  
4 }
```

```
1 public class Aclass {  
2     private int x;  
3     protected int y;  
4  
5     public Aclass (int a, int b) {  
6         x = a;  
7         y = b;  
8     }  
9  
10    public int addEm () {  
11        return x + y;  
12    }  
13  
14    public String toString () {  
15        return "" + x + " " + y;  
16    }  
17 }
```

