PRACTICE EXAM 1

The real exam has 60 questions and you are given three hours. It's the same with this exam.

1. Given:

```
public class Bang extends Thread {
  static Thread t1, t2, t3;
  public static void main(String[] args) throws Exception {
    t1 = new Thread(new Bang());
    t2 = new Thread(new Bang());
    t3 = new Thread(new Bang());
    t1.start();   t2.start();   t3.start();
  }
  public void run() {
    for(int i = 0; i < 500; i++) {
      System.out.print(Thread.currentThread().getId() + " ");
      if(i == 250)
        try {
          System.out.print("**" + t1.getId() + "**");
          t1.sleep(600);
        }
        catch (Exception e) { }
} } }
```

Which are true? (Choose all that apply.)

A. Compilation fails.

B. An exception is thrown at runtime.

C. Bang will execute for a second or two.

D. Bang will execute for at least 10 minutes.

E. `Thread t1` will almost certainly be the last thread to finish.

F. `Thread t1` will almost certainly be the first thread to finish.

G. It's difficult to predict which thread will be the last to finish.

2. Given:

```
public class Dec26 {
  public static void main(String[] args) {
    short a1 = 6;
    new Dec26().go(a1);
    new Dec26().go(new Integer(7));
  }
  void go(Short x) { System.out.print("S "); }
  void go(Long x) { System.out.print("L "); }
  void go(int x) { System.out.print("i "); }
  void go(Number n) { System.out.print("N "); }
}
```

What is the result?

A. `i L`
B. `i N`
C. `S L`
D. `S N`
E. Compilation fails.
F. An exception is thrown at runtime.

3. Given:

```
 1. public class Fellowship {
 2.   public static void main(String[] args) {
 3.     // insert code here
 4.   }
 5. }
 6. class Numinor {
 7.   enum Members {
 8.     HOBBITS(48), ELVES(74), DWARVES(50);
 9.     int height;
10.     Members(int h) { height = h; }
11.     int getHeight() { return height; }
12.   }
13. }
```

And these four lines of code to be inserted, independently at line 3:

```
I.    int h0 = Numinor.Members.HOBBITS.getHeight();
II.   int h1 = Numinor.Members.getHeight();
III.  int h2 = Members.HOBBITS.getHeight();
IV.   int h3 = Members.height;
```

Which are true? (Choose all that apply.)

A. Line I will compile.
B. Line II will compile.
C. Line III will compile.
D. Line IV will compile.
E. Class `Numinor` will NOT compile.

4. Given:

```
2. public class Volume {
3.   Volume v;
4.   int size;
5.   public static void main(String[] args) {
6.     Volume myV = new Volume();
```

```
 7.     final Volume v2;
 8.     v2 = myV.doStuff(myV);
 9.     v2.v.size = 7;
10.     System.out.print(v2.size);
11.   }
12.   Volume doStuff(Volume v3) {
13.     v3.size = 5;
14.     v3.v = new Volume();
15.     return v3;
16. } }
```

What is the result? (Choose all that apply.)

A. 5

B. 7

C. Compilation fails due to an error on line 8.

D. Compilation fails due to an error on line 9.

E. Compilation fails due to an error on line 13.

F. Compilation fails due to an error on line 14.

5. Given:

```
public class BirdHouse {
  public static void main(String[] args) {
    String r = "0";
    int x = -1, y = -5;
    if(x < 5)
    if(y > 0)
    if(x > y)
    r += "1";
    else r += "2";
    else r += "3";
    else r += "4";
    System.out.println(r);
} }
```

What is the result?

A. 0

B. 01

C. 02

D. 03

E. 013

F. 023

G. Compilation fails.

6. Given:

```
class c1 { }
class c2 { }
interface i1 { }
interface i2 { }
class A extends c2 implements i1 { }
class B implements i1 implements i2 { }
class C implements c1 { }
class D extends c1, implements i2 { }
class E extends i1, i2 { }
class F implements i1, i2 { }
```

What is the result? (Choose all that apply.)

A. Class `A` does not compile.

B. Class `B` does not compile.

C. Class `C` does not compile.

D. Class `D` does not compile.

E. Class `E` does not compile.

F. Class `F` does not compile.

G. Compilation succeeds for all of the classes.

7. Given that `"it, IT"` and `"pt"` are valid `Locale` codes, and given:

```
41.   Date d = new Date();
42.   DateFormat df;
43.   Locale[] la = {new Locale("it", "IT"), new Locale("pt")};
44.   for(Locale l: la) {
45.     df = DateFormat.getDateInstance(DateFormat.FULL, l);
46.     System.out.println(d.format(df));
47.   }
```

Which are true? (Choose all that apply.)

A. An exception will be thrown at runtime.

B. Compilation fails due to an error on line 43.

C. Compilation fails due to an error on line 45.

D. Compilation fails due to an error on line 46.

E. Classes from the `java.text` package are used in this code.

F. Classes from the `java.util` package are used in this code.

8. Given:

```
 2. class SuperCool {
 3.   static String os = "";
 4.   void doStuff() { os += "super "; }
 5. }
 6. public class Cool extends SuperCool {
 7.   public static void main(String[] args) {
 8.     new Cool().go();
 9.   }
10.   void go() {
11.     SuperCool s = new Cool();
12.     Cool c = (Cool)s;
13.     // insert code here
14.   }
15.   void doStuff() { os += "cool "; }
16. }
```

If the rest of the code compiles, which line(s) of code, inserted independently at line 13, compile? (Choose all that apply.)

A. `c.doStuff();`

B. `s.doStuff();`

C. `this.doStuff();`

D. `super.doStuff();`

E. `c.super.doStuff();`

F. `s.super.doStuff();`

G. `this.super.doStuff();`

H. There are other errors in the code.

9. Given:

```
 5.  static String s = "";
 6.  public static void main(String[] args) {
 7.    try { doStuff(args); }
 8.    catch (Error e) { s += "e "; }
 9.    s += "x ";
10.    System.out.println(s);
11.  }
12.  static void doStuff(String[] args) {
13.    if(args.length == 0)
14.      throw new IllegalArgumentException();
15.    s += "d ";
16.  }
```

And, if the code compiles, and given a `java` invocation with no arguments, what is the result? (Choose all that apply.)

A. `d x`

B. `e x`

C. `d e x`

D. Compilation fails due to an error on line 8.

E. Compilation fails due to an error on line 12.

F. Compilation fails due to an error on line 14.

G. An uncaught `IllegalArgumentException` is thrown.

10. Given:

```
 2. class Paratrooper implements Runnable {
 3.   public void run() {
 4.     System.out.print(Thread.currentThread().getName() + " ");
 5. } }
 6. public class Jump {
 7.   static Paratrooper p;
 8.   static { p = new Paratrooper(); }
 9.   { Thread t1 = new Thread(p, "bob"); t1.start(); }
10.   public static void main(String[] args) {
11.     new Jump();
12.     new Thread(new Runnable() { public void run()
          { ; }}, "carol").start();
13.     new Thread(new Paratrooper(), "alice").start();
14.   }
15.   Jump() { Thread t2 = new Thread(p, "ted"); t2.start(); }
16. }
```

Which are true? (Choose all that apply.)

A. The output could be `ted bob alice`

B. The output could be `bob alice carol`

C. The output could be `bob carol ted alice`

D. Compilation fails due to an error on line 8.

E. Compilation fails due to an error on line 9.

F. Compilation fails due to an error on line 12.

G. Compilation fails due to an error on line 15.

11. Use the fragments below to fill in the blanks so that the code will compile, and when invoked with:

```
java Dropkick fish
```

will produce the output: `"1 4 "`

Note: All of the blanks must be filled, not all the fragments will be used, and fragments can be used only once.

Code:

```
public class Dropkick {
  public static void main(String[] args) {
    boolean test = false;
    String[] s = {"duck", null, "frog"};

    if((s[1] == null) ___ (s[1].length() == 0)) System.out.print("1 ");

    if((s[2] == null) ___ (test ___ true)) System.out.print("2 ");

    if((s[0].equals("duck")) ___ (args[0].equals("fish")))
       System.out.print("3 ");
    if((args[0] != null) && (______)) System.out.print("4 ");
  }
}
```

Fragments:

```
|, ||, &, &&, ^, <, %, =, ==, !=, false, test, s[1]
```

12. Which are true? (Choose all that apply.)

A. For a specific object, it's NOT possible for `finalize()` to be invoked more than once.

B. It's possible for objects, on whom `finalize()` has been invoked by the JVM, to avoid the GC.

C. Overriding `finalize()` ensures that objects of that type will always be GCed when they become eligible.

D. The `finalize()` method is invoked only for GC-eligible objects that are NOT part of "islands of isolation."

E. For every object that the GC considers collecting, the GC remembers whether `finalize()` has been invoked for that specific object.

13. Given that:

`Exception` is the superclass of `IOException` and

`IOException` is the superclass of `FileNotFoundException`

and

```
2. import java.io.*;
3. class Author {
4.   protected void write() throws IOException { }
5. }
```

```
 6. public class Salinger extends Author {
 7.   private void write(int x) { }
 8.   protected void write(long x) throws FileNotFoundException { }
 9.   protected void write(boolean x) throws Exception { }
10.   protected int write(short x) { return 7; }
11.   public void write() { }
12. }
```

What is the result? (Choose all that apply.)

A. Compilation succeeds.

B. Compilation fails due to an error on line 7.

C. Compilation fails due to an error on line 8.

D. Compilation fails due to an error on line 9.

E. Compilation fails due to an error on line 10.

F. Compilation fails due to an error on line 11.

14. Given:

```
class Chilis {
  Chilis(String c, int h) { color = c; hotness = h; }
  String color;
  int hotness;
  public boolean equals(Object o) {
    if(this == (Chilis)o) return true;
    return false;
  }
  public String toString() { return color + " " + hotness; }
}
```

If instances of class `Chilis` are to be used as keys in a `Map`, which are true? (Choose all that apply.)

A. Without overriding `hashCode()`, the code will not compile.

B. As it stands, the `equals()` method has been legally overridden.

C. It's possible for such keys to find the correct entries in the `Map`.

D. It's NOT possible for such keys to find the correct entries in the `Map`.

E. As it stands, the `Chilis` class legally supports the `equals()` and `hashCode()` contracts.

F. If `hashCode()` was correctly overridden, it would make retrieving `Map` entries by key easier.

15. Given:

```
2. public class Contact {
3.   private String name;
4.   private String city;
5.   String getName() { return name; }
```

```
  void setName(String n) { name = n; }
  void setCity(String c) {
    if(c == null) throw new NullPointerException();
    city = c;
  }
  String getCity() { return city; }
}
```

Which are true? (Choose all that apply.)

A. Compilation fails.

B. The class is well encapsulated.

C. The `setCity()` method is an example of loose coupling.

D. The `setCity()` method has better encapsulation than `setName()`.

E. The `setCity()` method is cohesive; the `setName()` method is not.

16. Given the current directory is `bigApp`, and the directory structure:

```
bigApp
    |-- classes
            |-- Cloned.class
```

And the file:

```
public class Cloned {
  public static void main(String[] args) {
    System.out.println("classes");
    assert(Integer.parseInt(args[0]) > 0);
} }
```

Which will produce the output `"classes"` followed by an `AssertionError`? (Choose all that apply.)

A. `java -cp classes Cloned -4`

B. `java -cp classes -ea Cloned`

C. `java -ea-cp classes Cloned -4`

D. `java -ea -cp classes Cloned 4`

E. `java -ea, cp classes Cloned 4`

F. `java -ea -cp classes Cloned -4`

G. `java -cp classes Cloned -4 -ea`

17. Given:

```
interface Syrupable {
  void getSugary();
}
```

```
 4. abstract class Pancake implements Syrupable { }
 5.
 6. class BlueBerryPancake implements Pancake {
 7.   public void getSugary() { ; }
 8. }
 9. class SourdoughBlueBerryPancake extends BlueBerryPancake {
10.   void getSugary(int s) { ; }
11. }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.

B. Compilation fails due to an error on line 2.

C. Compilation fails due to an error on line 4.

D. Compilation fails due to an error on line 6.

E. Compilation fails due to an error on line 7.

F. Compilation fails due to an error on line 9.

G. Compilation fails due to an error on line 10.

18. Given:

```
 1. public class Endless {
 2.   public static void main(String[] args) {
 3.     int i = 0;
 4.     short s = 0;
 5.     for(int j = 0, k = 0; j < 3; j++) ;
 6.     for(int j = 0; j < 3; counter(j)) ;
 7.     for(int j = 0, int k = 0; j < 3; j++) ;
 8.     for(; i < 5; counter(5), i++) ;
 9.     for(i = 0; i < 3; i++, System.out.print("howdy ")) ;
10.   }
11.   static int counter(int y) { return y + 1; }
12. }
```

What is the result? (Choose all that apply.)

A. `howdy howdy howdy`

B. The code runs in an endless loop.

C. Compilation fails due to an error on line 5.

D. Compilation fails due to an error on line 6.

E. Compilation fails due to an error on line 7.

F. Compilation fails due to an error on line 8.

G. Compilation fails due to an error on line 9.

19. Given:

```
 2. class Big {
 3.   void doStuff(int x) { }
 4. }
 5. class Heavy extends Big {
 6.   // void doStuff(byte b) { }
 7.   // protected void doStuff(int x) throws Exception { }
 8. }
 9. public class Weighty extends Heavy {
10.   // void doStuff(int x) { }
11.   // String doStuff(int x) { return "hi"; }
12.   // public int doStuff(int x) { return 7; }
13.   // private int doStuff(char c) throws Error { return 1; }
14. }
```

Which method(s), if uncommented independently, compile? (Choose all that apply.)

A. Line 6

B. Line 7

C. Line 10

D. Line 11

E. Line 12

F. Line 13

20. Which are true? (Choose all that apply.)

A. A given `TreeSet`'s ordering cannot be changed once it's created.

B. The `java.util.Properties` class is conceptually more like a List than like a Map.

C. It's a reasonable design choice to use a `LinkedList` when you want to design queue-like functionality.

D. Of the main types of collections flavors (Lists, Sets, Maps), Queues are conceptually most like Sets.

E. It's programmatically easier to perform a non-destructive traversal of a `PriorityQueue` than a `LinkedList`.

F. Classes that implement the `Set` interface are usually well suited for applications that require access to a collection based on an index.

21. Given the following directory structure:

```
test -|
      | - Finder.class
      | - testdir -|
                   | - subdir
                   | - subdir2
                   | - testfile.txt
```

If `test`, `testdir`, `subdir`, and `subdir2` are all directories, and `Finder.class` and `testfile.txt` are files, and given:

```
import java.io.*;
public class Finder {
  public static void main(String[] args) throws IOException {
    String[] files = new String[100];
    File dir = new File(args[0]);
    files = dir.list();
    System.out.println(files.length);
} }
```

And, if the code compiles, the invocation:

```
java Finder testdir
```

What is the result?

A. `1`

B. `2`

C. `3`

D. `4`

E. `5`

F. `100`

G. Compilation fails.

H. An exception is thrown at runtime.

22. Given:

```
1. public class Grids {
2.   public static void main(String[] args) {
3.     int [][] ia2;
4.     int [] ia1 = {1,2,3};
5.     Object o = ia1;
6.     ia2 = new int[3][3];
7.     ia2[0] = (int[])o;
8.     ia2[0][0] = (int[])o;
9. } }
```

What is the result? (Choose all that apply.)

A. Compilation fails due to an error on line 4.

B. Compilation fails due to an error on line 5.

C. Compilation fails due to an error on line 6.

D. Compilation fails due to an error on line 7.

E. Compilation fails due to an error on line 8.

F. Compilation succeeds and the code runs without exception.

G. Compilation succeeds and an exception is thrown at runtime.

23. Given:

```
 3. public class OffRamp {
 4.   public static void main(String[] args) {
 5.     int [] exits = {0,0,0,0,0,0};
 6.     int x1 = 0;
 7.
 8.     for(int x = 0; x < 4; x++) exits[0] = x;
 9.     for(int x = 0; x < 4; ++x) exits[1] = x;
10.
11.     x1 = 0; while(x1++ < 3) exits[2] = x1;
12.     x1 = 0; while(++x1 < 3) exits[3] = x1;
13.
14.     x1 = 0; do { exits[4] = x1; }  while(x1++ < 7);
15.     x1 = 0; do { exits[5] = x1; }  while(++x1 < 7);
16.
17.     for(int x: exits)
18.       System.out.print(x + " ");
19. } }
```

What is the result?

A. `3 3 2 2 6 6`

B. `3 3 3 2 7 6`

C. `3 3 3 2 7 7`

D. `4 3 3 2 7 6`

E. `4 3 3 2 7 7`

F. Compilation fails.

24. Given:

```
 2. import java.util.*;
 3. public class HR {
 4.   public static void main(String[] args) {
 5.     List<Integer> i = new Vector<Integer>();
 6.     i.add(3); i.add(2); i.add(5);
 7.     int ref = 1;
 8.     doStuff(ref);
 9.     System.out.println(i.get(ref));
10.   }
11.   static int doStuff(int x) {
12.     return ++x;
13. } }
```

What is the result?

A. 2

B. 3

C. 5

D. Compilation fails.

E. An exception is thrown at runtime.

25. Given:

```
 2. import java.util.*;
 3. public class Vinegar {
 4.   public static void main(String[] args) {
 5.     Set<Integer> mySet = new HashSet<Integer>();
 6.     do1(mySet, "0");  do1(mySet, "a");
 7.     do2(mySet, "0");  do2(mySet, "a");
 8.   }
 9.   public static void do1(Set s, String st) {
10.     s.add(st);
11.     s.add(Integer.parseInt(st));
12.   }
13.   public static void do2(Set<Integer> s, String st) {
14.     s.add(st);
15.     s.add(Integer.parseInt(st));
16. } }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.

B. Compilation fails due to an error on line 6.

C. Compilation fails due to an error on line 13.

D. Compilation fails due to an error on line 14.

E. Compilation fails due to an error on line 15.

F. If only the line(s) of code that don't compile are removed, the code will run without exception.

G. If only the line(s) of code that don't compile are removed, the code will throw an exception.

26. Given:

```
3. class Employee {
4.   private String name;
5.   void setName(String n) { name = n; }
6.   String getName() { return name; }
7. }
8. interface Mungeable {
9.   void doMunging();
10. }
11. public class MyApp implements Mungeable {
12.   public void doMunging() { ; }
13.   public static void main(String[] args) {
14.     Employee e = new Employee();
15.     e.setName("bob");
16.     System.out.print(e.getName());
17. } }
```

Which are true? (Choose all that apply.)

A. MyApp is-a Employee.

B. MyApp is-a Mungeable.

C. MyApp has-a Employee.

D. MyApp has-a Mungeable.

E. The code is loosely coupled.

F. The Employee class is well encapsulated.

27. Given that `FileNotFoundException` extends `IOException`, and given:

```
 2. import java.io.*;
 3. public class MacPro extends Laptop {
 4.   public static void main(String[] args) {
 5.     new MacPro().crunch();
 6.   }
 7.   // insert code here
 8. }
 9. class Laptop {
10.   void crunch() throws IOException { }
11. }
```

Which method(s), inserted independently at line 7, compile? (Choose all that apply.)

A. `void crunch() { }`

B. `void crunch() throws Exception { }`

C. `void crunch(int x) throws Exception { }`

D. `void crunch() throws RuntimeException { }`

E. `void crunch() throws FileNotFoundException { }`

28. Given:

```
 2. class Horse {
 3.   String hands = "15";
 4. }
 5. class GaitedPony extends Horse {
 6.   static String hands = "14";
 7.   public static void main(String[] args) {
 8.     String hands = "13.2";
 9.     String result = new GaitedPony().getSize(hands);
10.     System.out.println(" " + result);
11.   }
12.   String getSize(String s) {
13.     System.out.print("hands: " + s);
14.     return hands;
15. } }
```

What is the result?

A. `14`

B. `15`

C. `hands: 13.2 14`

D. `hands: 13.2 15`

E. Compilation fails.

F. An exception is thrown at runtime.

29. Given:

```
 2. public class Humping {
 3.   public static void main(String[] args) {
 4.     String r = "-";
 5.     char[] c = {'a', 'b', 'c', 'z'};
 6.     for(char c1: c)
 7.       switch (c1) {
 8.         case 'a': r += "a";
 9.         case 'b': r += "b"; break;
10.         default: r += "X";
11.         case 'z': r+= "z";
12.       }
13.     System.out.println(r);
14. } }
```

What is the result?

A. `-abXz`

B. `-abbXz`

C. `-abbXzz`

D. `-abbXzXz`

E. Compilation fails due to a single error.

F. Compilation fails due to multiple errors.

30. Given:

```
import java.util.*;
public class Garage {
  public static void main(String[] args) {
    Map<String, String> hm = new HashMap<String, String>();
    String[] k = {null, "2", "3", null, "5"};
    String[] v = {"a", "b", "c", "d", "e"};

    for(int i=0; i<5; i++) {
      hm.put(k[i], v[i]);
      System.out.print(hm.get(k[i]) + " ");
    }
    System.out.print(hm.size() + " " + hm.values() + "\n");
 } }
```

What result is most likely?

A. `a b c a e 4 [c, b, a, e]`

B. `a b c d e 4 [c, b, a, e]`

C. `a b c d e 4 [c, d, b, e]`

D. `a b c`, followed by an exception.

E. An exception is thrown with no other output.

F. Compilation fails due to error(s) in the code.

31. Given:

```
class Jiggy extends Thread {
  Jiggy(String n) { super(n); }
  public void run() {
    for(int i = 0; i < 100; i++) {
      if("t1".equals(Thread.currentThread().getName()) && i == 5) {
        new Jiggy("t3").start();
        throw new Error();
      }
      if("t2".equals(Thread.currentThread().getName()) && i == 5) {
        new Jiggy("t4").start();
        throw new Error();
      }
      System.out.print(Thread.currentThread().getName() + "-");
    }
  }
  public static void main(String[] args) {
    Thread t1 = new Jiggy("t1");
    Thread t2 = new Jiggy("t2");
    t1.setPriority(1);  t2.setPriority(9);
    t2.start();  t1.start();
} }
```

Which are true? (Choose all that apply.)

A. Compilation fails.

B. After throwing error(s), `t3` will most likely complete before `t4`.

C. After throwing error(s), `t4` will most likely complete before `t3`.

D. The code will throw one error and then no more output will be produced.

E. The code will throw two errors and then no more output will be produced.

F. After throwing error(s) it's difficult to determine whether `t3` or `t4` will complete first.

32. Given:

```
class Stereo { void makeNoise() { assert false; } }
public class BoomBox extends Stereo {
  public static void main(String[] args) {
    new BoomBox().go(args);
  }
  void go(String[] args) {
    if(args.length > 0)  makeNoise();
    if(!args[0].equals("x")) System.out.println("!x");
} }
```

And, if the code compiles, the invocation:

```
java -ea BoomBox
```

What is the result?

A. `!x`

B. Compilation fails.

C. An `AssertionError` is thrown.

D. A `NullPointerException` is thrown.

E. An `IllegalArgumentException` is thrown.

F. An `ArrayIndexOutOfBoundsException` is thrown.

33. Given:

```
public class LaSelva extends Beach {
  LaSelva() { s = "LaSelva"; }
  public static void main(String[] args) { new LaSelva().go(); }
   void go() {
    Beach[] ba = { new Beach(), new LaSelva(), (Beach) new LaSelva() };
    for(Beach b: ba)  System.out.print(b.getBeach().s + " ");
  }
  LaSelva getBeach() { return this; }
}
class Beach {
```

```
11.   String s;
12.   Beach() { s = "Beach"; }
13.   Beach getBeach() { return this; }
14. }
```

What is the result?

A. `Beach LaSelva Beach`

B. `Beach LaSelva LaSelva`

C. `Beach LaSelva` followed by an exception.

D. Compilation fails due to an error at line 5.

E. Compilation fails due to an error at line 6.

F. Compilation fails due to an error at line 8.

G. Compilation fails due to an error at line 13.

34. When using the `java.io.Console` class, which are true? (Choose all that apply.)

A. Objects of type `java.io.Console` are created using a constructor from the same class.

B. Objects of type `java.io.Console` are created using a method from the `java.io.File` class.

C. Objects of type `java.io.Console` are created using a method from the `java.lang.System` class.

D. Objects of type `java.io.Console` are created using a method from the `java.lang.Object` class.

E. The method(s) designed to read passwords can optionally disable the echoing of user input.

F. The method(s) designed to read passwords return a `char[]`.

35. Given:

```
 3. public class Stealth {
 4.   public static void main(String[] args) {
 5.     Integer i = 420;
 6.     Integer i2;
 7.     Integer i3;
 8.     i2 = i.intValue();
 9.     i3 = i.valueOf(420);
10.     System.out.println((i == i2) + " " + (i == i3));
11. } }
```

What is the result?

A. `true true`

B. `true false`

C. `false true`

D. `false false`
E. Compilation fails.
F. An exception is thrown at runtime.

36. Given:

```
import java.io.*;
interface Risky {
  String doStuff() throws Exception;
  Risky doCrazy();
  void doInsane();
}
class Bungee implements Risky {
  public String doStuff() throws IOException {
    throw new IOException();
  }
  public Bungee doCrazy() { return new Bungee(); }
  public void doInsane() throws NullPointerException {
    throw new NullPointerException();
} }
```

What is the result? (Choose all that apply.)

A. Compilation succeeds.
B. The `Risky` interface will not compile.
C. The `Bungee.doStuff()` method will not compile.
D. The `Bungee.doCrazy()` method will not compile.
E. The `Bungee.doInsane()` method will not compile.

37. Given that `IllegalArgumentException` extends `RuntimeException`, and given:

```
static String s = "";
public static void main(String[] args) {
  try { doStuff(); }
  catch (Exception ex) { s += "c1 "; }
  System.out.println(s);
}
static void doStuff() throws RuntimeException {
  try {
    s += "t1 ";
    throw new IllegalArgumentException();
  }
  catch (IllegalArgumentException ie) { s += "c2 "; }
  throw new IllegalArgumentException();
}
```

What is the result?

A. `c1 t1 c2`

B. `c2 t1 c1`

C. `t1 c1 c2`

D. `t1 c2 c1`

E. Compilation fails.

F. An uncaught exception is thrown at runtime.

38. Given:

```
public class Networking {
  public static void main(String[] args) {
    List<Integer> i = new LinkedList<Integer>();
    i.add(4); i.add(2); i.add(5);
    int r = 1;
    doStuff(r);
    System.out.println(i.get(r));
  }
  static int doStuff(int x) {
    return ++x;
} }
```

What is the result?

A. `2`

B. `4`

C. `5`

D. Compilation fails.

E. An exception is thrown at runtime.

39. Given:

```
import java.util.*;
public class Unturned {
  public static void main(String[] args) {
    String[] towns = {"aspen", "vail", "t-ride", "dillon"};
    MySort ms = new MySort();
    Arrays.sort(towns, ms);
    System.out.println(Arrays.binarySearch(towns, "dillon"));
  }
  static class MySort implements Comparator<String> {
    public int compare(String a, String b) {
      return b.compareTo(a);
} } }
```

What is the most likely result?

A. `-1`

B. `1`

C. `2`

D. `3`

E. Compilation fails.

F. An exception is thrown at runtime.

40. Given:

```
class Weed {
  protected static String s = "";
  final void grow() { s += "grow "; }
  static final void growFast() { s += "fast "; }
}
public class Thistle extends Weed {
  void grow() { s += "t-grow "; }
  void growFast() { s+= "t-fast "; }
}
```

Which are the FEWEST change(s) required for this code to compile? (Choose all that apply.)

A. `s` must be marked `public`.

B. `Thistle.grow()` must be marked `final`.

C. `Weed.grow()` must NOT be marked `final`.

D. `Weed.growFast()` must NOT be marked `final`.

E. `Weed.growFast()` must NOT be marked `static`.

F. `Thistle.growFast()` must be removed from the class.

41. Given:

```
import java.util.regex.*;
public class Decaf {
  public static void main(String[] args) {
    Pattern p = Pattern.compile(args[0]);
    Matcher m = p.matcher(args[1]);
    while(m.find())
      System.out.print(m.group() + " ");
} }
```

And the three command-line invocations:

I. `java Decaf "0([0-7])?" "1012 0208 430"`

II. `java Decaf "0([0-7])*" "1012 0208 430"`

III. `java Decaf "0([0-7])+" "1012 0208 430"`

Which are true? (Choose all that apply.)

A. All three invocations will return valid octal numbers.

B. None of the invocations will return valid octal numbers.

C. Only invocations II and III will return valid octal numbers.

D. All three invocations will return the same set of valid octal numbers.

E. Of those invocations that return only valid octal numbers, each invocation will return a different set of valid octal numbers.

42. Given:

```
class Locker extends Thread {
  private static Thread t;
  public void run() {
    if(Thread.currentThread() == t) {
      System.out.print("1 ");
      synchronized(t) { doSleep(2000); }
      System.out.print("2 ");
    } else {
      System.out.print("3 ");
      synchronized(t) { doSleep(1000); }
      System.out.print("4 ");
    }
  }
  private void doSleep(long delay) {
    try { Thread.sleep(delay); } catch(InterruptedException ie) { ; }
  }
  public static void main(String args[]) {
    t = new Locker();
    t.start();
    new Locker().start();
} }
```

Assuming that `sleep()` sleeps for about the amount of time specified in its argument, and that all other code runs almost instantly, which are true? (Choose all that apply.)

A. Compilation fails.

B. An exception could be thrown.

C. The code could cause a deadlock.

D. The output could be `1 3 4 2`

E. The output could be `1 3 2 4`

F. The output could be `3 1 4 2`

G. The output could be `3 1 2 4`

43. Fill in the blanks using the fragments below, so that the code compiles and produces the output: `"1 3 2 3 2 "`

Note: You might not need to fill in all of the blanks. Also, you won't use all of the fragments, and each fragment can be used only once.

Code:

```
interface Gadget { }
class Watch {
  class Workings implements Gadget {
    Workings() ____________________________

    void tick() ______________________________

    ________________________    _____________________
  }
  _____________________  ______________________
}
public class Timer {
  public static void main(String[] args) {
    Watch w = new Watch();

    __________________ ww = w.new Workings();

    __________________
} }
```

Fragments:

```
{ System.out.print("2 "); }            w.tick();
{ Workings in = new Workings(); }      Watch()
{ System.out.print("3 "); }            Watch.Workings
{ System.out.print("1 "); }            Workings
ww.tick();                  Workings()
w.Workings                  void tock()
```

44. Given:

```
 2. public class Later {
 3.   public static void main(String[] args) {
 4.     boolean earlyExit = new Later().test1(args);
 5.     if(earlyExit) assert false;
 6.     new Later().test2(args);
 7.   }
 8.   boolean test1(String[] a) {
 9.     if (a.length == 0) return false;
```

```
10.      return true;
11.    }
12.    private void test2(String[] a) {
13.      if (a.length == 2) assert false;
14.  } }
```

Which are true? (Choose all that apply.)

A. Compilation fails.

B. The assertion on line 5 is appropriate.

C. The assertion on line 13 is appropriate.

D. `"java -ea Later"` will run without error.

E. `"java -ea Later x"` will run without error.

F. `"java -ea Later x y"` will run without error.

G. `"java -ea Later x y z"` will run without error.

45. Given:

```
String s = "1234";
StringBuilder sb =
  new StringBuilder(s.substring(2).concat("56").replace("7","6"));
System.out.println(sb.append("89").insert(3,"x"));
```

What is the result?

A. `34x5689`

B. `345x689`

C. `345x789`

D. `23x45689`

E. `23x45789`

F. Compilation fails.

46. Given the following pseudo-code design for a new accounting system:

```
class Employee
  maintainEmployeeInfo()
  connectToRDBMS()

class Payroll
  setStateTaxCodes()
  findEmployeesByState()

class Utilities
  getNetworkPrinter()
```

Assuming the class and method names provide good definitions of their own functionalities, which are probably true? (Choose all that apply.)

A. These classes appear to have low cohesion.

B. These classes appear to have high cohesion.

C. These classes appear to have weak validation.

D. These classes appear to have strong validation.

E. These classes appear to have weak encapsulation.

F. These classes appear to have strong encapsulation.

47. Given:

```
 2. class Dog {
 3.   void makeNoise() { System.out.print("bark "); }
 4.   static void play() { System.out.print("catching "); }
 5. }
 6. class Bloodhound extends Dog {
 7.   void makeNoise() { System.out.print("howl "); }
 8.   public static void main(String[] args) {
 9.     new Bloodhound().go();
10.     super.play();
11.     super.makeNoise();
12.   }
13.   void go() {
14.     super.play();
15.     makeNoise();
16.     super.makeNoise();
17. } }
```

What is the result? (Choose all that apply.)

A. `catching howl bark catching bark`

B. `catching howl howl catching howl`

C. `catching howl bark`, then an exception.

D. Compilation fails due to an error on line 10.

E. Compilation fails due to an error on line 11.

F. Compilation fails due to an error on line 14.

48. Given:

```
3. public class Baskin {
4.   public static void main(String[] args) {
5.     int i = 4;
6.     int j = 1;
7.
```

```
    assert(i > Integer.valueOf(args[0]));
    assert(j > Integer.valueOf(args[0])): "error 1";
    assert(j > i): "error 2": "passed";
} }
```

And, if the code compiles, given the following two command-line invocations:

I. `java -ea Baskin 2`

II. `java -ea Baskin 0`

Which are true? (Choose all that apply.)

A. Compilation fails.

B. Invocations I and II will throw an `AssertionError` that will add `String` data to the program's execution log.

C. Invocations I and II will throw an `AssertionError` that will add `String` data to the program's stack trace.

D. Not all of the `assert` statements use assertions appropriately.

49. Given:

```
import java.util.*;
class Priorities {
  public static void main(String[] args) {
    PriorityQueue toDo = new PriorityQueue();
    toDo.add("dishes");
    toDo.add("laundry");
    toDo.add("bills");
    toDo.offer("bills");
    System.out.print(toDo.size() + " " + toDo.poll());
    System.out.print(" " + toDo.peek() + " " + toDo.poll());
    System.out.println(" " + toDo.poll() + " " + toDo.poll());
} }
```

What is the result?

A. `3 bills dishes laundry null null`

B. `3 bills bills dishes laundry null`

C. `3 dishes dishes laundry bills null`

D. `4 bills bills dishes laundry null`

E. `4 bills bills bills dishes laundry`

F. `4 dishes laundry laundry bills bills`

G. Compilation fails.

H. An exception is thrown at runtime.

50. Given that the working directory is `bigApp`, and the following directory structure:

```
bigApp
    |-- classes
    |       |-- com
    |            |-- wickedlysmart
    |-- source
            |-- com
                 |-- wickedlysmart
                              |-- BigAppClass2.java
```

And the code:

```
1. public class BigAppClass2 { int doMore() { return 17; } }
```

And the following command-line invocations:

I. `javac -d source/com/wickedlysmart/BigAppClass2.java`

II. `javac -d classes source/com/wickedlysmart/BigAppClass2.java`

III. `javac -d classes/com/wickedlysmart source/com/wickedlysmart/BigAppClass2.java`

Which are true? (Choose all that apply.)

A. Invocation I will compile the file and place the .class file in the `bigApp` directory.

B. Invocation II will compile the file and place the .class file in the `classes` directory.

C. Invocation I will compile the file and place the .class file in the `wickedlysmart` directory.

D. Invocation II will compile the file and place the .class file in the `wickedlysmart` directory.

E. Invocation III will compile the file and place the .class file in the `wickedlysmart` directory.

51. Given:

```
 1. class Contact {
 2.   String doStuff() { return "howdy "; }
 3. }
 4. class Supplier extends Contact {
 5.   String doStuff() { return "send money "; }
 6.   public static void main(String[] args) {
 7.     Supplier s1 = new Supplier();
 8.     Contact c3 = new Contact();
 9.     Contact c1 = s1;
10.     Supplier s2 = (Supplier) c1;
11.     Supplier s3 = (Supplier) c3;
12.     Supplier s4 = new Contact();
13. } }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.

B. The code runs without exception.

C. If the line(s) of code that do NOT compile (if any) are removed, the code runs without exception.

D. If the line(s) of code that do NOT compile (if any) are removed, the code throws an exception at runtime.

52. Given that `Integer.parseInt()` throws `NumberFormatException`, and given:

```
 3. public class Ladder {
 4.   public static void main(String[] args) {
 5.     try {
 6.       System.out.println(doStuff(args));
 7.     }
 8.     catch (Exception e) { System.out.println("exc"); }
 9.     doStuff(args);
10.   }
11.   static int doStuff(String[] args) {
12.     return Integer.parseInt(args[0]);
13. } }
```

And, if the code compiles, given the invocation:

```
java Ladder x
```

What is the result? (Choose all that apply.)

A. `0`

B. `exc`

C. `"exc"` followed by an uncaught exception.

D. Compilation fails due to an error on line 4.

E. Compilation fails due to an error on line 9.

F. Compilation fails due to an error on line 11.

G. An uncaught exception is thrown with no other output.

53. Given the proper imports and given:

```
81.    String in = "1234,77777,689";
82.    Scanner sc = new Scanner(in);
83.    sc.useDelimiter(",");
84.    while(sc.hasNext())
85.      System.out.print(sc.nextInt() + " ");
86.    while(sc.hasNext())
87.      System.out.print(sc.nextShort() + " ");
```

What is the result?

A. `1234 77777 689`

B. Compilation fails.

C. `1234 77777 689 1234 77777 689`

D. `1234` followed by an exception.

E. `1234 77777 689` followed by an exception.

F. `1234 77777 689 1234` followed by an exception.

54. Given:

```
1. public class Glank implements Vonk { Jooker[] j; }
2. abstract class Bostron { String yoodle; Bostron b; }
3. interface Protefor { }
4. interface Vonk extends Protefor { int x = 7; }
5. class Jooker { Bostron b; }
```

Which are true? (Choose all that apply.)

A. Glanks have a Bostron.

B. Jookers implement Protefors.

C. Glanks implement Bostrons.

D. Jookers have a String.

E. Bostrons implement Vonks.

F. Bostrons have a Bostron.

55. Given that the root directory contains a subdirectory called `"office"` that contains some files for a Java application, if `"X"` and `"Y"` are unknown arguments, and the following command is invoked from the root directory in order to create a JAR file containing the `office` directory:

```
jar -cf X Y
```

Which are true? (Choose all that apply.)

A. `X` should be the file name of the JAR file, and `Y` should be `"office"`.

B. `X` should be `"office"`, and `Y` should be the file name of the JAR file.

C. Specifying a file name of the JAR file here is optional.

D. If a file name is not specified here, a file named `office.jar` will be created.

E. The file name, if specified, must be ended with `.jar` extension.

F. It is required that the `"office"` directory must initially have a subdirectory called `"META-INF"`.

G. All of the files other than .java and .class files must be initially placed in the `META-INF` directory.

56. Given a partial API:

Final class `Items` implements no interfaces and has one constructor:

```
Items(String name, int value)
```

And given that you want to make collections of `Items` objects and sort them (using classes and interfaces in `java.lang` or `java.util`), sometimes by `name`, and sometimes by `value`, which are true? (Choose all that apply.)

A. It's likely that you'll use the `Arrays` class.

B. It's likely that you'll use the `Collections` class.

C. It's likely that you'll implement `Comparable` at least twice.

D. It's likely that you'll implement `Comparator` at least twice.

E. It's likely that you'll implement the `compare()` method at least twice.

F. It's likely that you'll implement the `compareTo()` method at least twice.

57. Given:

```
 1. import java.util.*;
 2. public class Drunken {
 3.   public static void main(String[] args) {
 4.     Set<Stuff> s = new HashSet<Stuff>();
 5.     s.add(new Stuff(3));  s.add(new Stuff(4)); s.add(new Stuff(4));
 6.     s.add(new Stuff(5));  s.add(new Stuff(6));
 7.     s = null;
 8.     // do more stuff
 9.   }
10. }
11. class Stuff {
12.   int value;
13.   Stuff(int v) { value = v; }
14. }
```

When line 8 is reached, how many objects are eligible for garbage collection?

A. 4

B. 5

C. 6

D. 8

E. 10

F. 12

58. Given:

```
 1. public class Hose <E extends Hose> {
 2.   E innerE;
 3.   public static E doStuff(E e, Hose<E> e2) {
 4.     // insert code here
 5.   }
 6.   public E getE() {
 7.     return innerE;
 8. } }
```

Which can be inserted, independently at line 4, for the code to compile? (Choose all that apply.)

A. `return e;`

B. `return e.getE();`

C. `return e2;`

D. `return e2.getE();`

E. `return new Hose().getE();`

F. Compilation fails regardless of which return is inserted.

59. Given the following method signatures from `ArrayList`:

```
boolean add(E e)
protected void removeRange(int fromIndexInclusive, int toIndexExclusive)
int size()
```

and given:

```
 2. import java.util.*;
 3. public class MyUtil extends ArrayList {
 4.   public static void main(String[] args) {
 5.     MyUtil m = new MyUtil();
 6.     m.add("w");  m.add("x");  m.add("y");  m.add("z");
 7.     m.removeRange(1,3);
 8.     System.out.print(m.size() + " ");
 9.     MyUtil m2 = new MyUtil2().go();
10.     System.out.println(m2.size());
11.   }
12. }
13. class MyUtil2 {
14.   MyUtil go() {
15.     MyUtil m2 = new MyUtil();
16.     m2.add("1");  m2.add("2");  m2.add("3");
17.     m2.removeRange(1,2);
18.     return m2;
19. } }
```

What is the result?

A. `1 1`

B. `1 2`

C. `2 1`

D. `2 2`

E. An exception is thrown at runtime.

F. Compilation fails due to a single error.

G. Compilation fails due to multiple errors.

60. Given:

```
public class Hug implements Runnable {
  static Thread t1;
  static Hold h, h2;
  public void run() {
    if(Thread.currentThread().getId() == t1.getId()) h.adjust();
    else h2.view();
  }
  public static void main(String[] args) {
    h = new Hold();
    h2 = new Hold();
    t1 = new Thread(new Hug());
    t1.start();
    new Thread(new Hug()).start();
} }
class Hold {
  static int x = 5;
  synchronized void adjust() {
    System.out.print(x-- + " ");
    try { Thread.sleep(200); } catch (Exception e) { ; }
    view();
  }
  synchronized void view() {
    try { Thread.sleep(200); } catch (Exception e) { ; }
    if(x > 0) adjust();
} }
```

Which are true? (Choose all that apply.)

A. Compilation fails.

B. The program could deadlock.

C. The output could be `5 4 3 2 1`

D. The program could produce thousands of characters of output.

E. If the `sleep()` invocations were removed the chances of deadlock would decrease.

F. If the `view()` method was not synchronized the chances of deadlock would decrease.