

CSE132 Midterm Sample Exam Questions

NOTE: This is intended to give you an idea of the **format** of the midterm exam. This sample is **not** representative of the complete coverage of the exam. See the review sheet for an overview of the exam content. The actual exam will have 52 questions, and will cover more topics.

1. Object references are passed to a method
 - a. by value, so the caller and the method share the same object.
 - b. by reference, so the method can change the pointer in the caller.
 - c. by copy, so the caller and the method work with separate objects.
 - d. by value-result, so the changes are copied back to the caller afterwards.
 - e. All of the above.
 - f. None of the above.

2. When a method is *overloaded*, the choice of which method to call is determined
 - a. at run-time, based on the types of the objects.
 - b. at compile-time, using the most specific method that matches the parameter expression types.
 - c. at run-time, based on which method would not throw an exception.
 - d. at compile-time, based on the which method has a return type that will correctly type-check in the expression making the call.
 - e. at compile-time if possible, but otherwise at run-time.
 - f. None of the above.

3. The cast ((Foo) x)
 - a. changes the type of x to Foo.
 - b. can be prevented with the **protected** access modifier.
 - c. is necessary when calling methods polymorphically.
 - d. can be used to call a method defined in Foo on object x.
 - e. All of the above.
 - f. None of the above.

4. Suppose Bar is the return type of the method get(x). The cast ((Foo) get(x))
 - a. is unnecessary if Bar extends Foo.
 - b. would throw a ClassCastException if get(x) returns an object not of type Foo.
 - c. wouldn't compile if Foo and Bar have the same parent class.
 - d. would *not* throw an exception if get(x) returns null.
 - e. All of the above.
 - f. None of the above.

5. An adapter is often extended in order to
 - a. implement an interface without writing all the methods.
 - b. create an anonymous class.
 - c. listen for user events.
 - d. Both (a) and (c).
 - e. All of the above.
 - f. None of the above.

6. Items in the application programmer interface (API)
 - a. should never be used outside the package.
 - b. must be **public**.
 - c. may be either **public** or **protected**.
 - d. should never throw exceptions.
 - e. All of the above.
 - f. None of the above.

Items 7-16 refer to the following class definitions. For items 7-15, indicate whether it is **true** or **false** or **cannot be determined** from the information given. For item 16, provide a short answer.

```
public class Bar extends Foo {
    int max;
    static LinkedList contents = new LinkedList();
    public Bar(int max) throws Exception {
        if (max < 0)
            throw new IllegalArgumentException("max is negative");
        this.max = max;
    }
    public void add(Foo f, int m) {
        max = Math.max(m, max);
        if (contents.size() >= max)
            throw new RuntimeException("no space left");
        contents.add(f);
    }

    public static void main(String[] args) throws Exception {
        Bar b = new Bar(0);
        b.add(b, 1);
    }
}
```

7. If **Foo** declares an instance variable **max** of type **int**, the **add** method in **Bar** would use the inherited variable instead of the one declared in **Bar**.
8. Any method that creates an instance of **Bar** must either do so in a **try** block or declare that it throws an exception.
9. Any method that calls the **add** method must either do so in a **try** block or declare that it throws an exception.
10. The class would not compile if the words “throws Exception” were removed from the constructor declaration.
11. When the **add** method is called, variable **m** is stored on the stack.
12. When the **add** method is called, the object that **f** refers to is stored on the heap, but the variable **f** is on the stack.
13. If three instances of **Bar** are created, three **LinkedList** objects will be created.
14. The following code fragment would execute without throwing an exception.
 Bar b = new **Bar**(0);
 b.add(b,1);
15. If the word **static** were deleted from the declaration of **contents**, the class hierarchy would permit instances of **Bar** to represent containers that may contain other such containers inside of them.
16. Assuming only one instance of **Bar** is ever created, state a representation invariant that is maintained by **Bar**.

A weighted graph is an abstraction consisting of some number vertices and various edges that connect pairs of vertices. For example, in a graph representing an airlines flight map, the vertices of graph would represent cities and each edge could represent the cost of an airline ticket to travel directly between the two vertices it connects. The following implementation is undocumented. Deduce the functionality from the code in order to answer the questions below. *Assume the code compiles without error.*

```
public final class DenseGraph implements Graph {
    int[ ][ ] edges;
    DenseGraph(int numVertices) {
        edges = new int[numVertices][numVertices];
    }
    void set(int vertex1, int vertex2, int weight) {
        edges[vertex1][vertex2] = weight;
    }
    int get(int vertex1, int vertex2) {
        return edges[vertex1][vertex2];
    }
    int indirect(int vertex1, int vertex2) {
        int min = get(vertex1, vertex2);
        for (int i = 0; i < edges.length; i++)
            if (get(vertex1, i) > 0 && get(i, vertex2) > 0)
                min = Math.min(min, get(vertex1, i) + get(i, vertex2));
        if (min == 0)
            throw new NoSuchPathException(vertex1 + " to " + vertex2);
        return min;
    }
}
```

17. What is the best choice of access modifier(s) for the `DenseGraph` constructor?
 18. What is the best choice of access modifier(s) for the array `edges`?
 19. What is the best choice of access modifier(s) for the method named `indirect`?
 - 20-22. Provide REQUIRES, MODIFIES, and EFFECTS documentation for the `set` method.
 - 23-26. Provide REQUIRES, MODIFIES, and EFFECTS, and RETURNS documentation for the `indirect` method.
 27. Is `NoSuchPathException` a descendant of `RuntimeException`?
 28. Explain your answer to 27.
 29. If your REQUIRES clause for the `set` method were violated, what would happen?
 30. What code could you insert at the beginning of the `set` method to provide the user with better feedback when the REQUIRES clause for the `set` method is violated?
 31. Assuming `set` is always called with a positive weight, what might go wrong in user code if the `indirect` method simply returned 0 instead of throwing an exception?
- BONUS: Provide an abstraction function for the set of edges represented by instances of the `DenseGraph` class.