CS 251
Intermediate Programming
Quiz 6

Brooke Chenoweth

University of New Mexico

Spring 2025

Question 1: What's wrong with this code?

```
public static long foo(int n) {
  boolean do = false;
  return do ? n*2 : 0;
}
```

A Cannot use a keyword as a variable name.

B Invalid operator on return line.

C Return value does not match method return type.

D Did not initialize value of n.

E Some other error.

F This code will successfully compile.

Question 1: What's wrong with this code?

```
public static long foo(int n) {
  boolean do = false;
  return do ? n*2 : 0;
}
```

A Cannot use a keyword as a variable name.

B Invalid operator on return line.

C Return value does not match method return type.

D Did not initialize value of n.

E Some other error.

F This code will successfully compile.

Question 2: What's wrong with this code?

```
public class MyClass {
    private int x = 10;

    public static void main(String[] args) {
        x++;
        System.out.println(x);
    }
}
```

A The println method expects String, not int.
B Can't access private variable x from a public method.
C Can't access x without an instance of MyClass.
D Variable x is a constant, so can't be incremented in main.
E Some other error.
F This code will successfully compile.

Question 2: What's wrong with this code?

```
public class MyClass {
    private int x = 10;

    public static void main(String[] args) {
        x++;
        System.out.println(x);
    }
}
```

A The println method expects String, not int.
B Can't access private variable x from a public method.
C Can't access x without an instance of MyClass.
D Variable x is a constant, so can't be incremented in main.
E Some other error.
F This code will successfully compile.

Question 3: What's wrong with this code?

```
public static void bar(int n) {
  if(5 > n) {
    System.out.println("Small");
  } else {
    throw new Exception("Big!");
  }
}
```

A Invalid test in if statement.

B Did not declare method throws an exception.

C No value returned in method.

D Did not initialize value of n.

E Some other error.

F This code will successfully compile.

Question 3: What's wrong with this code?

```
public static void bar(int n) {
  if(5 > n) {
    System.out.println("Small");
  } else {
    throw new Exception("Big!");
  }
}
```

A Invalid test in if statement.

B Did not declare method throws an exception.

C No value returned in method.

D Did not initialize value of n.

E Some other error.

F This code will successfully compile.

Question 4: Keywords

What is the keyword used to
make a class that cannot be extended?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
make a class that cannot be extended?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
access a member variable hidden by a local variable?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
access a member variable hidden by a local variable?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
call the parent class version of a method overridden in the child class?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
call the parent class version of a method overridden
in the child class?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- **super**
- this

Question 4: Keywords

What is the keyword used to
make a class that cannot be instantiated?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
make a class that cannot be instantiated?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
make a variable or method belong to a *class* rather than an *instance*?

- abstract
- class
- default
- final
- private
- protected
- public
- static
- super
- this

Question 4: Keywords

What is the keyword used to
make a variable or method belong to a *class* rather than an *instance*?

- abstract
- class
- default
- final
- private
- protected
- public
- **static**
- super
- this

Question 5: True or False?

Every class has a default no-argument constructor.

Question 5: True or False?

Every class has a default no-argument constructor.

- False
- If you don't create a constructor yourself, the class will have a do-nothing no-argument constructor, but if you define any constructor at all, you will no longer have this constructor provided for you
- If you want to still have a no-argument constructor, you will have to explicitly define one.

Question 6: True or False?

An abstract class cannot contain any method implementations.

Question 6: True or False?

An abstract class cannot contain any method implementations.

- False
- An abstract class may (but does not have to) contain method implementations.
- You may want to partially implement a class in an abstract parent class and then finish the implementation in a concrete child class.

Question 7: True or False?

In the expression `3 > 4 && foo(5)`, method `foo` will be called with an argument of 5.

Question 7: True or False?

In the expression `3 > 4 && foo(5)`, method `foo` will be called with an argument of 5.

- False
- Short circuit evaluation of logical operators means that if the value of the entire expression can be determined from the value of left hand expression, the right hand expression will not be evaluated at all.

Question 8

Which of the following does *not* correctly declare and instantiate a map that associates `String` keys with `Double` values?

A

```
Map<String, Double> map = new HashMap<String, Double>();
```

B

```
HashMap<String, Double> map = new HashMap<String, Double>();
```

C

```
Map<String, Double> map = new TreeMap<String, Double>();
```

D `Map<String, Double> map = new HashMap<>();`

E `Map<String, Double> map = new TreeMap<>();`

F `Map<String, Double> map = new Map<>();`

G All of the above.

H None of the above.

Question 8

Which of the following does *not* correctly declare and instantiate a map that associates `String` keys with `Double` values?

A
```
Map<String, Double> map = new HashMap<String, Double>();
```
B
```
HashMap<String, Double> map = new HashMap<String, Double>();
```
C
```
Map<String, Double> map = new TreeMap<String, Double>();
```
D `Map<String, Double> map = new HashMap<>();`

E `Map<String, Double> map = new TreeMap<>();`

F `Map<String, Double> map = new Map<>();`

G All of the above.

H None of the above.

Question 9

What is the value of the following expression?

```
1 + "2"+ 3 + 4 * 5 + 6
```

A `123456`
B `21`
C `42`
D `"1229"`
E `"123206"`
F `"123456"`
G `"1241"`
H `"1266"`
I Some other value.
J The value of this expression is undefined.
K This expression would result in a compilation error.

Question 9

What is the value of the following expression?

```
1 + "2"+ 3 + 4 * 5 + 6
```

A `123456`
B `21`
C `42`
D `"1229"`
E `"123206"`
F `"123456"`
G `"1241"`
H `"1266"`
I Some other value.
J The value of this expression is undefined.
K This expression would result in a compilation error.

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`
- All the `+` operators have equal precedence, so now we evaluate from left to right.

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`
- All the `+` operators have equal precedence, so now we evaluate from left to right.
- `"12" + 3 + 20 + 6`

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`
- All the `+` operators have equal precedence, so now we evaluate from left to right.
- `"12" + 3 + 20 + 6`
- `"123" + 20 + 6`

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`
- All the `+` operators have equal precedence, so now we evaluate from left to right.
- `"12" + 3 + 20 + 6`
- `"123" + 20 + 6`
- `"12320"+ 6`

Question 9: Reasoning

```
1 + "2"+ 3 + 4 * 5 + 6
```

- The `*` operator has higher precedence than than `+`
- `1 + "2"+ 3 + 20 + 6`
- All the `+` operators have equal precedence, so now we evaluate from left to right.
- `"12" + 3 + 20 + 6`
- `"123" + 20 + 6`
- `"12320"+ 6`
- `"123206"`

Question 10

Which methods can access the protected members of a class?

A Only static methods of the same class.
B Only final methods of the same class.
C Only protected methods of the same class.
D Only private and protected methods of the same class.
E Only methods defined in the same class.
F Only methods defined in the same package (including those in the class itself).
G Only methods within the same class or its children.
H Only methods within the same class, its children, or in the same package.

Question 10

Which methods can access the protected members of a class?

A Only static methods of the same class.
B Only final methods of the same class.
C Only protected methods of the same class.
D Only private and protected methods of the same class.
E Only methods defined in the same class.
F Only methods defined in the same package (including those in the class itself).
G Only methods within the same class or its children.
H Only methods within the same class, its children, or in the same package.

Question 11

Fill in the blanks to make a method that takes two arguments, a `Collection` of `String` objects and a `String` to search for, and returns a collection containing only the strings found in the input collection that contain the search string somewhere in them. The method should be able to be called without an instance of the containing class. You may assume that the arguments are not null and that the Collection does not contain any null objects. Do not modify the collection passed in as an argument.

Question 11

```
public BLANK BLANK
  filterStrings(BLANK strs,
                String search) {

  Collection<String> result = new BLANK;

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return BLANK;
}
```

Question 11

```
public static BLANK
  filterStrings(BLANK strs,
                String search) {

  Collection<String> result = new BLANK;

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return BLANK;
}
```

Question 11

```
public static Collection<String>
  filterStrings(BLANK strs,
                String search) {

  Collection<String> result = new BLANK;

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return BLANK;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new BLANK;

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return BLANK;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new ArrayList<>();

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return BLANK;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new ArrayList<>();

  for(BLANK) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return result;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new ArrayList<>();

  for(String s : strs) {
    BLANK

    if(s.contains(search)) {
      BLANK
    }
  }

  return result;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new ArrayList<>();

  for(String s : strs) {
    BLANK

    if(s.contains(search)) {
      result.add(s);
    }
  }

  return result;
}
```

Question 11

```
public static Collection<String>
  filterStrings(Collection<String> strs,
                String search) {

  Collection<String> result = new ArrayList<>();

  for(String s : strs) {
    /* no code here */

    if(s.contains(search)) {
      result.add(s);
    }
  }

  return result;
}
```