

Introductory Programming Midterm Exam

Consider the following code fragment:

```
public void answer(int an){  
    switch(an){  
        case 0:  
            System.out.print("Maybe");  
        case 1:  
            System.out.print("Yes");  
            break;  
        case 2:  
            System.out.print("No");  
    }  
}
```

What would be printed if 0 was given as a parameter

Select one:

1. What would be printed if 0 was given as a parameter?
 - a. "MaybeNo"
 - b. "MaybeYes"
 - c. "MaybeYesNo"
 - d. "Maybe"
 - e. No answer.

Consider the following code-fragment

```
public class Jeans extends Trousers{  
    public Jeans(){  
        super("Denim");  
    }  
}
```

What is the direct superclass of **Jeans**?

Select one:

2. What is the direct superclass of **Jeans**?
 - a. String
 - b. Object
 - c. Denim
 - d. Trousers
 - e. No answer.

Consider the following code:

```
Set<Integer> a = new HashSet<>();  
a.add(1);  
a.add(2);  
a.add(3);
```

What is the iteration order of the set **a**?

Select one:

3. What is the iteration order of the set **a**?
- a. 2, 3, 1
 - b. 3, 2, 1
 - c. The iteration order of a Set<T> is unspecified
 - d. 1, 2, 3
 - e. No answer.

Consider the following content of a file called "Main.java":

```
public class Main{  
    public static void main(String[] args){  
        Scanner in = new Scanner(System.in);  
        int n = in.nextInt();  
        String s = in.nextLine();  
    }  
}
```

The above code does not compile. What should you do to make the it compile?

4. The above code does not compile. What should you do to make the it compile?
- a. Add a try-catch block
 - b. Remove `int n = in.nextInt();`
 - c. Remove `String s = in.nextLine();`
 - d. Add `import java.util.Scanner;`
 - e. No answer.
5. How do you handle a checked exception in java?
- a. Checked means it is already handled by the compiler
 - b. An if-else statement
 - c. A switch-statement
 - d. A try-catch block
 - e. No answer.

Chekced exceptions can be considered as expection made by yourself.

6. If **A implements List** this means that?
- A can implement the methods of List it wants.
 - A has to implement all non-default methods of List.
 - A have to `@Override` at least one method from List.
 - A can use Lists as fields.
 - No answer.
7. If a method throws an exception, what happens to the return value?
- The catch block gest the return value as a parameter
 - The method does not return a value
 - The method returns the value but it will be null
 - The method returns the value, and when the try block is exited, the catch block is executed
 - No answer.
8. `Map<Stringm Integer> a = new Map<>();` will not compile because?
- Map requires a `Comparator<K>`
 - Map only works on primitive types
 - `new` is a reserved keyword in Java
 - Map is an interface and cannot be instantiated
 - No answer.
9. Suppose that **class A implements I**, that **class B extends A**, and that A, B and I each separately declare a method **m()**. Consider the code snippet. **B b = new b(); A a = b;**, **I i = a;** Upon invoking **I.m()**, which definition of m will be executed?
- Both A's and B's definition
 - B's definition
 - I's definition
 - A's definition
 - No answer.
10. Suppose that the file **A.java** starts. **public abstract class A implements I { public A(){...** An occurrence of **new A()** produces a compile error. Why?
- A implements an interface
 - The class A is not in scope
 - A is an abstract class
 - The access modifiers do not allow it
 - No answer

11. What always holds true if a primitive field is **final**?

- a. More than one method uses the field
- b. The field is static
- c. The field is immutable
- d. The class depends on the field
- e. No answer.

12. What does autoboxing mean?

- a. When the compiler automatically converts one primitive type into another
- b. When the compiler automatically collects elements in a new collection
- c. When the compiler automatically wraps a primitive value in a wrapper object
- d. When the compiler automatically generates a jar file of the executable program
- e. No answer.

13. What holds true for the following codesnippet: for (Ingredient i : ingredients)?

- a. Ingredient is a superclass of Iterator<Ingredient>
- b. Ingredient implements Iterator<Ingredient>
- c. i == ingredients
- d. ingredients is Iterable<Ingredient>

14. What is the definition of a Set<T> s in java?

- a. For some elements $e1 \neq e2$ is it the case that $e1.equals(e2)$
- b. If elements $e1 \neq e2$ are inserted at the same time, then $e1.equals(e2)$
- c. For no elements $e1 \neq e2$ is it the case that $e1.equals(e2)$
- d. For all elements $e1 \neq e2$ is it the case that $e1.equals(e2)$
- e. No answer.

15. What is the difference between an abstract class and an interface?

- a. An interface can extend any abstract class
- b. An interface can only extend an abstract class, it implements all the methods
- c. An abstract class can contain fields; An interface cannot
- d. An abstract class cannot implement an interface

16. What is the required relationship between the `.equals` and `.hashCode` methods?

- a. If `a.equals(b)` then `a.hashCode() == b.hashCode()`
- b. `a.hashCode() == b.hashCode()` if and only if `!a.equals(b)`
- c. `a.hashCode() == b.hashCode()` if and only if `a.equals(b)`
- d. if `a.hashCode() == b.hashCode()` then `a.equals(b)`
- e. No answer.

17. What is the **throws** keyword used for?

- a. To declare that a method might throw or propagate a given exception
- b. To throw an exception when something has gone wrong
- c. To define a new exception that can be used in the same package
- d. To check an exception
- e. No answer.

18. What is the type of **(new Integer(4)+38)**?

- a. String
- b. double
- c. int
- d. Integer
- e. No answer.

19. What should we aim for when designing classes?

- a. Low cohesion, low coupling
- b. High cohesion, low coupling
- c. Low cohesion, high coupling
- d. High cohesion, high coupling
- e. No answer.

Introductory Programming Midterm Resit Exam

20. What happens to `s.size()` when running `s.add(x)` method on a hashset `s`?
- a. No matter the value of `x` it is always unchanged
 - b. Depending on the value of `x` it is either unchanged or increases by one
 - c. No matter the value of `x` it is always increased by one
21. Java `String` objects have the method `public String concat(String str)` which returns the concatenation of two strings. Suppose `String firstname = "Harry"` and `String lastname="Potter"`, which of the following expressions returns the string `"Harry Potter"`?
- a. `Lastname.concat(firstname)`
 - b. `Firstname.concat(lastname)`
 - c. `Firstname.concat("").concat(lastname)`
 - d. `Lastname.concat("").concat(firstname)`
22. Consider the method. `public int myMethod(int a, int b) {if a≥b return a-b; else return b-a; }`. For testing `myMethod` which of the following assertions are suitable to be used in a test case (assuming the method is correct)?
- a. `assertTrue(myMethod(10,42)>0)`
 - b. `assertTrue(myMethod(20,39)==20)`
 - c. `assertEquals(11,myMethod(25,36))`
 - d. `assertEquals(myMethod(11,1),10)`
23. Which object types would be suitable (not necessarily best) for storing a song collection, allowing retrieval of a list of song titles by a band with a given name? Chose all that apply.
- a. `HashMap <List<String>, String>`
 - b. `TreeMap <List<String>, String>`
 - c. `HashMap <String, List<String>>`
 - d. `TreeMap <String, List<String>>`

Question 5

Not complete

Marked out of
8.00

🚩 Flag question

Consider the following class:

```
abstract class TestTaker {  
    public int points() {  
        return 20;  
    }  
  
    public final boolean pass() {  
        return points() / 5 > 9 && points() / 17 < 3;  
    }  
  
    public abstract String getName();  
}
```

Create a class Student that inherits from TestTaker, such that the example below compiles and prints "Passed!".

For example:

Test	Result
<pre>TestTaker taker = new Student(); if (taker.pass()) { out.println("Passed!"); } else { out.println("Failed!"); }</pre>	Passed!

24. Create a class Student that inherits from TestTaker, such that the example below compiles and prints "Passed!".

- Alternate the return value of point(), so that $\text{points()} / 5 > 9$, and $\text{points()} / 17 < 3$ points() should return 50;

25. Testing is most often applied to catch which type of errors?

- Syntax errors
- Logical errors
- Semantic errors

26. Which of the following are primitive types in Java?

- Double
- String
- boolean
- int

27. Which of the following kinds of variables can be accessed outside of a class?

- a. A private int field variable
- b. A private double field variable
- c. A public Boolean field variable
- d. A double local variable in a method
- e. An int local variable in a method

Question 9

Not complete

Marked out of 8.00

Flag question

Since the year 1700 the *Gregorian* calendar has been used in Denmark. In this calendar, most years have 365 days, but some have 366 days. A year most often has 366 days if it is divisible by 4. For example, the year 2020 has 366 days since the remainder $2020 \% 4$ (i.e., 2020 modulo 4) equals zero. There are some exceptions, though: If the year is divisible by 100 but not divisible by 400, it has 365 days even though it is divisible by 4. In this problem you must write a method

```
int numberOfDays(int year)
```

which returns the number of days in a given **year** (which can be assumed to be greater than 1700).

For example:

Test	Result
<code>System.out.println(numberOfDays(2020))</code>	366
<code>System.out.println(numberOfDays(2021))</code>	365
<code>System.out.println(numberOfDays(2000))</code>	366
<code>System.out.println(numberOfDays(1900))</code>	365

28.

Question 10

Not complete

Marked out of 8.00

Flag question

The following method adds 25% sales tax to a given amount, **beforeTax**:

```
double addSalesTax(double beforeTax) {  
    return beforeTax * 1.25;  
}
```

Write a method

```
double addTax(double beforeTax, double taxPercent)
```

that is able to add any desired percentage tax, and in particular **addTax(amount, 25)** should return the same result as **addSalesTax(amount)**.

For example:

Test	Result
<code>System.out.println(addTax(100,25))</code>	125.0
<code>System.out.println(addTax(100,12.5))</code>	112.5
<code>System.out.println(addTax(1000,100))</code>	2000.0

29.

30. Suppose **x** and **y** are integers. Which of the following boolean expressions are true if and only if **x** equals two times **y**?

- a. `x == 2 * y`
- b. `y == x / 2`
- c. `x / 2 == y`
- d. `y * 2 == x`

31. Consider the code snippet **Integer x = new Integer(1000); Integer y = new Integer(1000); int u = 1000; int v = 1000;** After running the above code, which of the following boolean expressions are true?

- a. `(y == u)`
- b. `(x == y)`
- c. `(x == u)`
- d. `(u == v)`

32. Which of the following java keywords can be used to handle special cases in the execution of a program?

- a. Return
- b. Catch
- c. Import
- d. final

33. Which of the following methods have the same signature as **public int foo(double d, int i)**?

- a. `public int foo(double d, int i)`
- b. `private int foo(Double d, Integer i)`
- c. `public int foo(int i, double d)`
- d. `public int boo(double d, int i)`
- e. `public Integer boo(double d, int i)`
- f. `private Integer foo(double d, int i)`

34. Consider the classes. **public class A{...}**, **public class B extends A{...}**, **public class C extends A{...}** Which of the following snippets results in a *compile-time* error?

- a. `B xb = new B();`
`C xc = new C();`
`A xa = (A)xc;`
- b. `B xb = new B();`
`A xa = xb;`
`C xc = new C();`
- c. `B xb = new B();`
`A xa = (A)xb;`
`C xc = (C)xb;`

35. Which of the following statements are correct?

- a. A good unit test should be isolated and run independently of other tests
- b. Identifying boundary conditions is not an important issue for unit testing
- c. In testing we should force error condition to be able to check whether the code can handle all such problems.
- d. We should include control flows in test cases

36. Which of these array literals satisfy **myArray.length==4**?

- a. `myArray = {0, 1, 2, 3};`
- b. `myArray = {1, 2, 3, 4};`
- c. `myArray = {4};`
- d. `myArray = {0, 1, 2, 3, 4};`

Question 6

Complete

Mark -1.00 out of 2.00

🚩 Flag question

Suppose that the method *sort* is implemented for sorting an array of numbers in ascending order. Assume in a test case the sort method is applied to **{1, 5, 8, 3, 6, 7}** and returns an array named *result*. Which one of the following assertions is suitable to be used in the test case for checking the correctness of the method?

Select one:

- ☒ a. `assertArrayEquals(new int[] {1, 3, 5, 6, 7, 8})`
- ☐ b. `assertArrayEquals(new int[] {1, 3, 5, 6, 7, 8}, result)`
- ☐ c. `assertArrayEquals(result, new int[] {1, 3, 5, 6, 7, 8})`
- ☐ d. `assertTrue(result, new int[] {1, 3, 5, 6, 7, 8})`

The correct answer is: `assertArrayEquals(new int[] {1, 3, 5, 6, 7, 8}, result)`

37. Which one of the following assertions is suitable to be used in the test case for checking the correctness of the method?
- a. `assertArrayEquals(new int[] {1, 3, 5, 6, 7, 8})`
 - b. `assertArrayEquals(new int[] {1, 3, 5, 6, 7, 8}, results)`
 - c. `assertArrayEquals(results, new int[] {1, 3, 5, 6, 7, 8})`
 - d. `assertTrue(result, new int[] {1, 3, 5, 6, 7, 8})`
38. Consider the method **public int myMethod(int a, int b) { if a+b≤10) b=b+2 else b=0; return b;}**. For testing myMethod with two test cases, which of the following sets of values of a and b are better inputs to use in the test cases?
- a. {a = 10, b = 2}, {a = 0, b = 0}
 - b. {a = 1, b = 0}, {a = 8, b = 2}
 - c. {a = 10, b = 0}, {a = 0, b = 10}
 - d. {a = 0, b = 10}, {a = 2, b = 10}
39. Which of the following statements can be used to create an object that supports the **Map** interface?
- a. `New HashMap();`
 - b. `New AbstractMap();`
 - c. `New Map();`
 - d. `New Treemap();`

40. Which of the following statements are correct?

- a. Each test method should be annotated with `@Test`
- b. We can use `@Before` (or `@BeforeEach`) for running a common code before each test case
- c. We can use `@After` (or `@AfterEach`) for running a common code after each class

41. Which of the following statements are true?

- a. A java program execution always starts at a main method
- b. Every class must have a main method
- c. The **main** method must be of type **public static void**
- d. The **main** method is run every time you create an object

42. Consider a method **public static Double addOne(Integer x) {...}**. Which statement is true (assuming no exception is cast)?

- a. `addOne(x)` can return an Integer or a Double
- b. `addOne(x)` always returns a Double
- c. `addOne(x)` always returns an Integer

Question **12**

Complete

Mark 2.00 out of 2.00

🚩 Flag question

Consider the code snippet:

```
HashSetInteger> s = new HashSetInteger>();  
s.add(0);  
s.add(1);  
s.add(2);  
s.add(1);
```

What is `s.size()` after running this code?

Answer:

The correct answer is: 3

43. Consider the code snippet above. What is the **s.size()** after running this code?

- a. Sets do not contain duplicates hence `s.size() = 3`;

44. Consider the code fragment **Integer a=1; Integer b=a; a=a+1; b=b+1;**. What are the values of **a** and **b** after executing this?
- a. Both a and b are equal to 3
 - b. Both a and b are equal to 2
 - c. B is equal to 2 and a is equal to 3
 - d. A is equal to 2 and b is equal to 3
45. Suppose **myMap** is the type **Map<Integer, List<String>>**. What is the type of **myMap.get(y)**?
- a. List<String>
 - b. Integer
 - c. String
 - d. Map<List<Integer>, Integer>
46. In what situation is it good practice to use the **protected** visibility modifier on a field?
- a. When other packages need to access the field
 - b. When using a private modifier results in compilation errors
 - c. When subclasses need to access the field
 - d. When a superclass needs to access the field
47. Consider classes **public class A {...}** and **public class B extends A {...}**. Which of the following statements are always true?
- a. Every method of class B is supported by class A
 - b. Objects of class A can be used in any context where objects of class B are allowed
 - c. Objects of class B can be used in any context where objects of class A are allowed.
 - d. Methods m of class A has the same functionality as method m of class B
48. Method overriding in Java is used to:
- a. Change the behavior of an object after it has been created
 - b. Support multiple methods with the same name
 - c. Make the behavior of a class different from the class it extends.