Exam 2 Study Guide

Feel free to work on this with other students, ask questions on piazza, and/or ask a TA for help. Note that this study guide does NOT cover everything that may be on the upcoming exam. The answer key will NOT be published, but students are encouraged to make their own.

1. True/False

1. \_\_\_\_\_\_ If you declare and instantiate a constant array: final int[] ARR = {1, 2};  
    You can change the second element by doing ARR[1] = 3;
   1. True, final just means you can’t reassign the reference that the variable points to. ARR = new int[] {2, 1}; is illegal. Note, if the assignment is an initialization within a constructor, it is legal, b/c construc is called only once. But in a method, it’s not. So for a class Meebo, public Meebo() {ARR = new int[] {2, 2};} is fine, but a method public void assign() {ARR = new int[] {2, 2};} is error
2. \_\_\_\_\_\_ The method signature consists of the return type, name, and parameter list.
   1. The signature doesn’t contain return type, only name and parameter types. This is because aslkdhkaycxiuhhhhhhhs
   2. Ex: public static double angle(double theta, String displayFormat); has signature angle(double,String)...that’s it (PROFESSOR OMOJOKUN PLEASE PETITION ORACLE TO CHANGE THIS)
3. \_\_\_\_\_\_ If I have a method:  
    public static boolean doSomething() {return true;}   
   Then:   
    public void doSomething(){}   
   will overload the previous method.
   1. NO NO BEEP BEEP BEEP YOU ARE BAD. Neither access modifiers, static, or return type are part of the method signature. WAHHH. So java will start crying if you put these two functions in a file
4. \_\_\_\_\_\_ If I have a method:   
    public String doSomething(String str, int num) {return “yes”;}  
   Then:  
    private int doSomething(int num, String str) {return num;}  
   will overload the previous method.
   1. Yes because the signatures are different: doSomething(String,int) vs doSomething(int,String)
5. \_\_\_\_\_\_ If I have a method:  
    public void doSomething(String str1, String str2) {}  
   Then:  
    public static void doSomething(String str2, String str1) {}  
   Will overload the previous method.
   1. No because even if the parameter *NAMES* are different, the *TYPES* are the same.
6. \_\_\_\_\_\_ You can overload a static method.
   1. YEE it is possible because even if the overloaded method behaves differently from the original, there is still only ONE version of that overloaded method, and it is shared between all classes.
7. \_\_\_\_\_\_In Java, every class you write is a subclass of at least one other class.
   1. Yes, because it either extends an existing class or automatically extends object.
8. \_\_\_\_\_\_In a constructor, if an explicit super call is present, it must be the first statement in the constructor.
   1. Yes. WHY?? WHY?? I don’t know.
9. \_\_\_\_\_\_If a class defines a single constructor, the constructor contains an implicit super call if no explicit super call is provided in the constructor.
   1. Consult <https://stackoverflow.com/questions/2054022/is-it-unnecessary-to-put-super-in-constructor>
   2. Basically, if the superclass has a constructor with NO ARGUMENTS that is ACCESSIBLE to the subclass, then ***yes*** that will be called, otherwise nope
10. \_\_\_\_\_\_You can define a subclass of an abstract class without defining any of the abstract methods defined in the superclass.
    1. Only if the subclass is also abstract. Otherwise you must override.
11. \_\_\_\_\_\_Overloading a superclass method in a subclass means defining a method with the same name as the superclass method but with a different parameter list.
12. \_\_\_\_\_\_Protected members are visible to classes in the same package and to subclasses.
13. \_\_\_\_\_\_Private members are visible in the class in which they are defined, but not in subclasses.

2: Multiple choice

1. How many of the following are true?
   1. You can only have abstract methods in abstract class.
      1. True if you’re talking about methods with the “abstract” keyword, but false because methods in interfaces are also abstract, just without the keyword
   2. Keywords like super and this can only be used in instance methods and not static methods.
      1. True because “this” is a reference to an instance of an object
   3. If a class you wrote doesn’t extends other classes, it means the class is the parent class of all and doesn’t inherit any methods.
      1. False
   4. If you have an @Override tag on a method that doesn’t override anything, java will give you a warning but still compiles your code.
      1. 1
      2. 2
      3. 3
      4. 4
2. How many of the following are true?
   1. concrete classes can extend abstract classes
   2. abstract classes can extend concrete classes
   3. subclasses of abstract classes must override all abstract methods from the superclass
   4. abstract classes cannot contain non-abstract methods
   5. the constructor of an abstract class can only be accessed using **super**, not **new**.
      1. 2
      2. 3
      3. 4
      4. 5
3. What package is the Object in the standard library located?
   1. Java.util
   2. Java.lang
   3. Object.java
   4. Object.lang
4. If I were to override the equals method in the object class in a School class, what would the method header be?
   1. public boolean equals(School other)
   2. public void equals(School other)
   3. public boolean equals (Object other)
   4. public void equals (Object other)
5. A method count() has the method signature below. Which of the following choices is a valid return type for a method that overrides count?
   1. public int count(float f){....}float
   2. Long
   3. Integer
   4. Object
   5. None of the above
6. Having getters and setters for a class is a demonstration of which of the following?
   1. Abstraction
   2. Inheritance
   3. Encapsulation
   4. Lossy Conversion
7. public boolean equals(Animal a) {

return true;

}

Is an example of

1. Overloading
2. Overriding
3. What access modifiers on class members allow subclasses to inherit them from superclass?
   1. public, no modifier, protected, private
   2. public, no modifier, protected
   3. public, no modifier
   4. public
   5. none of the above
4. Static methods are inherited
   1. True
   2. False

3. Tracing  
Given the following class definitions:

|  |
| --- |
| public class Super {  protected int x = 1;  public Super() {  System.out.print("Super");  }  }  public class Duper extends Super {  protected int y = 2;  public Duper() {  System.out.println(" duper");  }  }  public class Fly extends Super {  private int z, y;  public Fly() {  this(0);  }  public Fly(int n) {  z = x + y + n;  System.out.println(" fly times " + z);  }  public static void main(String[] args) {  Duper d = new Duper();  int delta = 1;  Fly f = new Fly(delta);  }  } |

What is printed when Fly is run?

Super duper

Super fly times 2

4. Short Answer

1. What is the difference between the keyword public and private?
2. What does the keyword static mean?
3. How would you run a java program called Test.java in the command line with arguments: 1, 3, 6.0?
4. How would you access these three elements inside your java program? What are their data types?
5. Say you have a method:

|  |
| --- |
| public static void doSomething(String str, int num){} |

Write another method which overloads this method.

1. Given this code for the class Dog, finish the constructor.

|  |
| --- |
| public class Dog {  private int age;  private String name;  private String breed;   public Dog(int age, String name, String breed) {      } |

1. Write the code to create a dog with an age of 11, breed of Husky, and name of Max and assign it to a variable called doggy.
2. Finish this method to make it compile and run:

|  |
| --- |
| public void woof() {    } |

1. This code has 2 compilation errors, write down the errors with explanations:

|  |
| --- |
| public abstract class Animal {  private int weight;  public abstract String makeNoise();  public int getWeight() {  return weight;  }  }  public class Lion extends Animal{  public int getWeight() {  return weight + 5;  }  }  public class Driver {  public static void main(String… args) {  Lion a = new Lion();  }  } |

Fill in the code for the following method that takes an array of numbers and returns an array of the eaven numbers in the array arguments.  
public int[] evens(int[] numbers) {  
 // Put your code below  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
}

Fill in the code for the following method that takes a String array which reverses the values within the array, so the element at index 0 is now at the last index, etc. Note that this method does not return anything.  
public void reverse(String[] arr) {  
 //Put your code below  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
}

Given this code, what are the 2 ways to access the numCreated value outside of this class?

|  |
| --- |
| public class Dog {  public static int numCreated = 0;  private String name;  public Dog(String name) {  this.name = name;  numCreated++;  }  public static void main(String[] args) {  Dog myDog = new Dog(“Buzz”);  } } |

Using Math’s static random method, write the code to generate a pseudo-random int value from [10, 150]. Remember that Math.random() returns a double value from [0,1).

Write a class named Square which has:

* 1. A constructor that takes in a String name and a double edgeLength.
  2. A constructor that takes in a double edgeLength and sets name to “”.
  3. A constructor that takes in nothing and sets name to “” and edgeLength to 1.0.
  4. A static method called isSquare that takes in 4 doubles to verify if the 4 doubles are equal. This method should return a boolean.
  5. A method that returns the area of the Square. Call this method area.
  6. A method that takes in another Square as a parameter and returns boolean whether they are equal to each other. Call this method equals.
  7. A method to change the name of the Square. Call this method setName.
  8. A method to change the edgeLength of the Square. Call this method setEdge.

Write a Temperature class that allows you to store a temperature type and temperature value (write constructor(s) that allow you to store values when you create objects of this class). Also, write a method changeUnit(String unit) to convert between different temperature units (for example, Celcius to Fahrenheit) and updates the values stored in the object. Write another method printTemp() to print out the values stored in the object. Finally, keep a track of the number of total Temperature objects you might have created (print this value when printTemp() is called).

Instance data:

* double temp; //contains the value of the temperature
* String units; //contains the units of the temperature (“C”, “F”, “R”, or “K”)
* static int count; //number of total Temperature objects created

Instance methods:

* void printTemp(); //prints out the current temperature as well as its units
* void changeUnit(String unit); //converts temperature to different units
* void setTemp(double temp); //sets the temperature
* double getTemp(); //returns the temperature
* double getUnit(); //returns the Unit
* static int getNumObjects(); //returns the number of Temperature objects created