CSDS 132 – Programming in Java
Midterm Summary Sheet
Charlie Lin (cxl1335), Emma Guarnieri (efg29)
March 2, 2024

Objectives:

- Understand how to use primitive and non-primitive types, and the differences between the two.
- Understand and write methods that make use of Java's arithmetic operations.
- Write classes with fields, constructors, and getter and setter methods.
- Overload methods and constructors.
- Override methods, especially those of `toString` and `equals`.
- Use if statements to solve complex problems.
- Given an API, create classes and methods.
- Manipulate Strings using for loops, while loops, and `StringBuilder`.

DISCLAIMER

If you find the problems and questions on here to be too challenging, don't be discouraged! This sheet is designed to (hopefully) be a little harder than the actual midterm exam, although we don't really know what will be on it! This is just an extra resource you can use to study for your midterm exam.

Also, we are covering a lot of material, but even this is not everything on the exam. Please go through the Midterm Review document and ask us any questions during the session!

Conceptual Questions

Primitive Types

1. Fill in the table below for primitive types.

Primitive Type	How many bits?	When do we use them?
double		
float		
long		
int		
char		
short		

byte		

2. Which of the following typecasts are automatic? Select all that apply.
    a. double → int
    b. float → long
    c. short → char
    d. boolean → char
    e. short → byte
    f. char → long

3. For Question 2, explain answers b, c, and d.

4. What is the resulting type and value of the following operations? The value ‘c’ has the int value of 99.
    a. `4 * 12`

    b. `(int) 3.0 / 4.0`

    c. `(double) 3 / 4`

    d. `(byte) 'c' + 200`

    e. `(byte) ('c' + 200)`

5. We are given the following statements in our program:
    a. `int x = 7;`
    b. `int y;`
    c. `int z;`
    d. `y = 8;`

    Which lines declare a variable? Which lines perform an assignment?

6. Suppose we have a function `idk()` that returns a boolean. When is this method evaluated, and when is it not evaluated?
    a. `true && idk()`
    b. `false && idk()`
    c. `true || idk()`
    d. `false || idk()`
    e. `false & idk()`
    f. `true | idk()`
    g. `!false && true || idk()`
    h. `!false & true || idk()`
    i. `!false && true | idk()`

7. Using your answers from 6, explain the difference between & vs. && and | vs. ||.

Non-Primitive Types and Object-Oriented Programming

8. Let us say we are given a class `Fruit`. There are 3 classes, `Berry, Banana,` and `Apple` which are subclasses of `Fruit`. Additionally, there are classes `Blueberry` and `Raspberry` which are subclasses of `Berry`. Which of the following declarations would be legal in Java?
    a. `Fruit a = new Fruit();`
    b. `Fruit b = new Berry();`
    c. `Fruit c = new Banana();`
    d. `Fruit d = new Raspberry();`
    e. `Berry e = new Blueberry();`
    f. `Banana f = new Blueberry();`
    g. `Apple g = new Fruit();`
    h. `Apple h = new Berry();`

9. Using the classes from Question 8, let us also say that `Fruit` has a method, `isReadyForHarvest`.
    a. Let's say within the `Fruit` class, there are two method headers for `isReadyForHarvest`:
        - `public boolean isReadyForHarvest() {…}`
        - `public boolean isReadyForHarvest(int season) {…}`

These headers allow you to either call `isReadyForHarvest` with no inputs or 1 integer input. What is the term used to describe this concept?

b. Let us say that `Berry` has its own implementation of `isReadyForHarvest`, which differs from `Fruit`'s implementation. What is the term used to describe this concept?

c. How would we call the `Fruit` version of `isReadyForHarvest` from `Berry`'s `isReadyForHarvest` method?

10. How should we check if primitive types are equal? What about non-primitive types?

11. What is the widest primitive type? Widest non-primitive type?

12. How do you call a static method? How do you call an instance/non-static method?

13. We are given a class, `MyClass`, with an instance field `num`. `MyClass` also has a static field, `staticNum`, initialized to 5.
    a. Let us say we create 2 instances of `MyClass`, `var1` and `var2`. We initialize `var1.num` = 6 and `var2.num` = 7. What is the result when we access `var1.num`? What happens when we access `var2.num`?

    b. Let us say there is a method in `MyClass`, "`setStaticNum`" that allows us to change the value of `staticNum` to a given value. Now let us say we call

`var2.setStaticNum(9)`. What is the result when we access `var1.staticNum`? What happens when we access `var2.staticNum`?

c. What best explains the answers for a. and b.?

14. List which modifiers (access modifiers, instance vs. static, and others) would be most appropriate.
    a. A method that gets a specific property of the object that we only want extending classes to see
    b. A constant in which all instances of a class will have the same value
    c. A helper method used in other methods of the instance, but shouldn't be accessed outside of the class

15. When is it possible to change the method that is called by typecasting the value to the left of the dot? Select all that apply.
    a. When the method is overridden
    b. When the method is overloaded.
    c. When the method is not static.
    d. When the method is static.

16. When is it absolutely necessary to overload a constructor?

17. In order to override a method, the method must be _____. (private/public/etc.? static/instance? Or does it matter?)

18. In order to override a method, what must be true about the return type?

Strings and Arrays

19. How do you get the size of an array? What about a String?

20. Can you change the length and/or content of a String? What about arrays?

21. How do you get the 4th character of a String? What about the 4th element of an Array?

22. What are the 2 ways to declare an array in Java?

23. Which of the following array creations is legal? If illegal, explain why it is illegal.
    a. `Berry[][][] berryCube = new Berry[10][][];`

    b. `Berry[][] grid = new Berry[][10];`

    c. `Berry[][][] berryCube2 = new Berry[20][10][100];`

24. Why do we say that Java doesn't technically have 2D arrays?

Coding Problems

Creating Classes Problems

25. Create a class `Hospital` that has 2 String values named `name` and `address` and an int value named `numEmployees`. These values should be provided on the instantiation of a class. Also make appropriate getters and setters. In addition, have 2 `Hospital` objects be "equal" if they have the same address and name. Finally, override the `toString` method so that it returns "Hospital <name> is located at <address>".

26. Create a class `Employee` that has 5 fields: `name` (a String); `hospital`, referring to the Hospital the employee works in; `hourlyPay` (a double); `hoursWorkedPerWeek` (a double); and `employeeID` (an int). Each employee ID should be unique across all hospitals, ascending, and start at 1, and automatically generated from the `Hospital` object by creating a `generateEmployeeID` method that returns an int in `Hospital`. (Hint: you might want to use `static` somewhere!). All of the fields, with the exception of `employeeID`, should be specified in the constructor and have getters and setters. In addition, `employeeID` should also have a getter method.
    Override the `toString` method so that it returns "<name> (<employeeID>) works at <hospital name>".
    In addition, have another method `getWeeklyPay` that returns the weekly pay, calculated as the hourly pay times the number of hours worked per week.

27. Create a class `Nurse` that extends `Employee`. Each `Nurse`, in addition to the fields of `Employee`, a `Nurse` has a double field `overtimePay`, and a double field `overtimeLimit`, which should be specified in the constructor and have appropriate getters and setters.
    `Nurse` must also be able to be instantiated without `overtimePay` or `overtimeLimit` in which `overtimePay` should be 1.5 times the `hourlyPay` and `overtimeLimit` should be 40.
    Override the `toString` method so that it returns "<name> (<employeeID>) works at <hospital name> as a nurse".
    In addition, the functionality of `getWeeklyPay` should be changed so that the weekly pay is calculated by multiplying the hourly pay times the number of regular hours, plus the number of overtime hours times the overtime pay. The number of maximum regular hours is specified as `overtimeLimit`.

28. Suppose we had the following API for `School`:
    `public class School extends Object`
    Constructor summary:
    - `School()`
    - `School(String name, double tuition, double teacherSalary, int numTeachers, int numStudents)`

    Method summary:
    - `double getTotalTuition()`: returns the tuition of all of the students.
    - `double getTeacherExpenses()`: returns all of the teachers' salaries.
    - `double getTeacherSalary()`: returns a single teacher's salary.
    - `void increaseTuition(int tuition)`: increases the tuition by blaming tuition hikes on "inflation."

- `void fireTeachers(int number)`:decreases the number of teachers.
- `void hireTeachers(int number)`: increases the number of teachers.

Create a new type of `School` called `University`. It should extend `School` and should work exactly the same as `School` except:

a. The constructor should take in a field named `president` that is a String.
b. There should be a method `getReputation` that returns an int. The method returns the reputation so far (saved as an appropriate field, default reputation should be 0).
c. The `hireTeachers` method should change. Teachers may only be hired if the hiring of teachers would result in the total profits of the university (e.g. tuition) still being greater than the total expenses of the university (e.g. teachers salaries). Otherwise, nothing will happen.

Writing Methods Problems

Put all the methods below in a class titled `Midterm.java`. Unless otherwise stated, you are NOT allowed to use any functions of Math for this problem.

29. Perfect numbers are positive integers that are equal to the sum of its divisors, not including itself. For instance, 6 is a perfect number because its divisors, 1, 2, and 3, all add up to make itself, 6. Write a method `isPerfectNumber` that takes in an integer and returns whether a number is perfect or not.

30. Write a method `largestPrimeDivisor` that takes in an int and returns an int which is the largest prime divisor of the number. You may assume that the number will be greater than 1.

31. Write a method `compoundInterest` that takes in 2 doubles and an int: the principal value, the annual interest rate in percent, and the number of years and returns a double, rounded to the hundredths place, of the balance with interest compounded. Remember that compound interest is computed using the following equation: $A = P(1 + r)^n$, where A is the final amount, P is the principal value, r is the interest rate to 2 decimal places, and n is the number of years.

32. **Challenge:** Write a method `calculateDay` that takes in 3 ints: the year, the month, and the day, and computes the day of the week, as an int (1 = Sunday, 2 = Monday, …). The day of the week can be programmatically calculated using the following equation:

$$W = (d + \lfloor(13 \times m - 1) / 5\rfloor + y + \lfloor y/4\rfloor + \lfloor c/4\rfloor - 2 \times c) \bmod 7$$

where W is the day of the week (0 = Sunday, 1 = Monday, …, 6 = Saturday; NOTE this is different from what we want to return), d is the day (1 to 31), m is the month (March = 1,

April = 2, …, February = 12), c is the century (for instance, 2020 would have a $c = 20$), and y is the last 2 digits of the year (for instance, 2015 would have a $y = 15$). The month is confusing; if we have a date of January 1st, 2000 we would give it a value of $m = 11$ and $y = 1999$. $\lfloor x \rfloor$ indicates we want to take the "floor" of the number x (for instance, the floor of 2.6 would be 2). *mod* 7 indicates the remainder of the division operation if we were to divide the number by 7. Note that this "remainder" can be negative, so we should flip it to make it positive. Assume that a proper date is put in (the above method only works after 1752).

String/StringBuilder Methods Problems

Put all the methods below in a class titled `Midterm.java`. You can only use the following methods in the API:

class String

- length
- charAt

class StringBuilder

- length
- charAt
- append
- toString

class Character

- any method

33. Write a method `capitalizeWords` that takes in a String and returns a String with each of the words capitalized. We define a word as a sequence of consecutive non-whitespace characters.

34. Write a method `swapUntil` that takes a String and three char values as input and returns a String, containing the contents of the String up until the 3rd char input is found or the end of the String. In addition, all of the chars that match the first char input should be replaced with the second char input, and vice versa. For instance, `swapUntil("connamacher is the best", 'c', 'n', 'e')` would return "noccamanh".

35. Write a method `sumOfDigits` that takes a String and returns an int representing the sum of all of the digits in the String. If there are no digits in the String, the method should return 0. For instance, `sumOfDigits("CSDS132 is the best")` would return 1+3+2=6.

36. **Challenge**: Write a method `incrementString` that takes a String and returns a String, containing the contents of the String but with all numbers shifted up by 1. For example, `incrementString("Your total is $3.49")` would become "Your total is $4.50" (9 would go to 0).

37. **Challenge**: Write a method `alternateCase` that takes a String and returns a String, and alternates the case of every other word, with the first word being uppercase. For example, alternateCase("EverYonE loVes jaVA") should return "EVERYONE loves JAVA."