# Faculty of Science
# COMP-202B - Foundations of Computing (Winter 2016) - All Sections
# Final Examination

April 21st, 2016
14:00 - 17:00

Examiners: Yang Cai [Section 1 TR (10:00-11:30)]
Jackie Chi Kit Cheung [Section 2 MW (10:00-11:30)]
Melanie Lyman-Abramovitch [Section 3 MWF (12:30-13:30)]

## Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** examination; only a letter-sized (8.5" by 11") **crib sheet** is permitted. This crib sheet can be single or double-sided; it can be handwritten or typed. Non-electronic translation dictionaries are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- Besides the above, only writing implements (pens, pencils, erasers, pencil sharpeners, etc.) are allowed. The possession of any other tools or devices is prohibited.
- Answer **all** true/false and multiple choice questions on the scantron sheet.
- Answer question 36 in the booklet provided.
- You may take your question sheet home. We recommend marking your answers down so that you may verify them against the posted solutions.
- This examination has **19** pages including this cover page, and is printed on both sides of the paper. On page 17, you will find information about **useful classes and methods**. **You may detach the Appendix from the examination if you wish.**
- The Examination Security Monitor Program detects pairs of students with unusually similar answer patterns on mulitple-choice exams. Data generated by this program can be used as admissible evidence, either to initiate or corroborate an investigation or a charge of cheating under Section 16 of the Code of Student Conduct and Disciplinary Procedures.
- **MAKE SURE TO WRITE YOUR NAME AND STUDENT ID ON THE SCANTRON AS WELL AS THE BOOKLET FOR THE LONG ANSWER PROBLEMS. PLEASE WRITE LEGIBLY**

### Scoring

The exam will be scored as follows:

1. Questions 1 to 10 are worth 1 point each
2. Questions 11 to 35 will be worth 2 points each
3. Questions 36 is worth 40 points

Version-1

## True/False Section (1 point each)

1. It is possible to access a `private` class attribute from outside of the constructor.

   (A) TRUE

   (B) FALSE

2. The following code snippet will compile:

```
final int[] x = {1,2,3,4,5};
x[0] = 12;
```

   (A) TRUE

   (B) FALSE

3. The `this` keyword can be used to over-ride the restrictions from the `private` access modifier.

   (A) TRUE

   (B) FALSE

4. You can create an instance of a class you have defined, even if you didn't write a constructor.

   (A) TRUE

   (B) FALSE

5. In Java, a non-static method can access a static variable in the same class.

   (A) TRUE

   (B) FALSE

6. An ArrayList in Java can store different types of values.

   (A) TRUE

   (B) FALSE

7. After the following code executes, the value of x[0] is 5.

```
public static void main(String[] args){
  int[] x = {1,2,3};
  method(x[0]);
}
public static void method(int a){
  a = 5;
}
```

   (A) FALSE

   (B) TRUE

8. We can use the == operator to compare two String objects. The result is always true if the two strings contain identical sequences of characters.

   (A) TRUE

   (B) FALSE

9. There must be exactly one catch block for each try block.

   (A) TRUE

   (B) FALSE

10. A `String` is a primitive type that comes by default with Java.

    (A) TRUE

    (B) FALSE

## Regular Multiple Choice (2 points each)

11. What does the following method do?

```
public static String method(String s, char a, char b){
  for(int i=0; i<s.length(); i++){
    if(s.charAt(i) == a){
        s.charAt(i) = b;
    }
  }
  return s;
}
```

(A) There is a compile-time error.

(B) Replaces all instances of the character a with the character b in the String s.

(C) Replaces all instances of the character ‘a’ with the character ‘b’ in the String s.

(D) Replaces the first instance of character a with character b in the String s.

(E) Replaces the first instance of character ‘a’ with character ‘b’ in the String s.

12. What does the following method do?

```
public static boolean method(String[] s, char a){
  boolean b1 = true;
  for(int i=0; i<s.length; i++){
    boolean b2 = false;
    for(int j=0; j<s[i].length(); j++){
      if(s[i].charAt(j)==a){
        b2 = true;
      }
    }
    b1 = b1 && b2;
  }
  return b1;
}
```

(A) Determines whether or not a character is in at least one String in an array of Strings

(B) Determines whether or not a character is in every String in an array of Strings

(C) Determines whether or not a character appears in exactly one String in an array of Strings

(D) Determines whether or not all Strings in an array of Strings contain only a single kind of character.

(E) Determines whether or not a character appears exactly once in each String in an array of Strings

13. **For questions 13-17, consult the Penguin class in the Appendix.**
Consider the following code:

```
Penguin p1 = new Penguin("Bobby", "emperor");
Penguin p2 = new Penguin("Brenda", "fairy");
p1.growUp(); p2.growUp();
Penguin p3 = Penguin.breed(p1, p2);
```

What is the height of Penguin p3?

(A) 8

(B) 10

(C) 7

(D) 60

(E) 33

14. Which of the following is a valid toString() method for this class? Recall that a valid toString() method is called automatically when we pass an object instance as input in a print statement.

(A)
```
public String toString(String s) { return s; }
```

(B)
```
public String toString() {
return p.name + " " + p.breed + " " + Penguin.height; }
```

(C)
```
public String toString() { return name + " " + breed + " " + height; }
```

(D)
```
public String toString() { return Penguin.name + " " + Penguin.breed +
" " + Penguin.height; }
```

(E)
```
public String toString(String s) { s = name + " " + breed + " " +
height;
return s; }
```

15. **Assume that the Penguin class now has a correct implementation of toString. (You can get points for the following problems, even if you got the previous question wrong).**
Consider the following block of code. What prints?

```
Penguin p1 = new Penguin("Norman", "fairy");
Penguin p2 = new Penguin("Lilly", "fairy");
p1.growUp();
Penguin p3 = p2.growUp();
System.out.println(p3);
System.out.println(p2);
```

(A) There is a compile-time error

(B) Lilly fairy 33
Lilly fairy 33

(C) Lilly fairy 7
Lilly fairy 7

(D) Lilly fairy 7
Lilly fairy 33

(E) Lilly fairy 33
Lilly fairy 7

16. Consider the following block of code. What prints?

```
Penguin p1 = new Penguin("Zain", "king");
Penguin p2 = new Penguin("Ursula", "emperor");
p1.growUp(); p2.growUp();
Penguin p3 = Penguin.breed(p1, p2);
p2 = p3;
p2.growUp();
System.out.println(p2);
System.out.println(p3);
```

(A) baby Mix 60.0
baby Mix 8.0

(B) Ursula emperor 100.0
baby Mix 8.0

(C) Ursula emperor 100.0
baby emperor 8.0

(D) Ursula emperor 100.0
baby emperor 100.0

(E) baby Mix 60.0
baby Mix 60.0

17. Consider the following block of code. What prints?

```
Penguin p1 = new Penguin("Zain", "king");
Penguin p2 = new Penguin("Ursula", "king");
p1.growUp();
Penguin p3 = Penguin.breed(p1, p2);
p3 = p1;
p1 = Penguin.breed(p3, p2.growUp());
System.out.println(p1);
```

(A) baby Mix 8.0

(B) baby king 8.0

(C) There is a compile-time error.

(D) null

(E) baby king 10.0

18. Consider the following block of code. What prints?

```
public class Abc{
  public int doSomething(int a,int b){return a;}
  public double doSomething(int a,int b){return (a+b);}
  public String doSomething(int a, int b){return (a+b);}

  public static void main(String args[]){
    Abc obj=new Abc();
    System.out.println(obj.doSomething(20,20));
  }
}
```

(A) 20

(B) There is a compile-time error.

(C) A string "40"

(D) 40

(E) 40.0

19. Examine the following code:

```
ArrayList<String> list = new ArrayList<String>() ;

list.add( "Andy" );
list.add( "Bart" );
list.add( "Carl" );
list.add( "Derek" )
list.add( 0, "Eve" );
```

What element will be at index 2 of the list?

(A) Carl

(B) Derek

(C) Eve

(D) Bart

(E) Andy

20. Examine the following code:

```
ArrayList<String> list = new ArrayList<String>() ;

list.add( "Andy" );
list.add( "Bart" );
list.add( "Carl" );
list.add( "Doug" );
list.add( "Elmo" );
```

Which of the following will change the list so that it looks like:

```
{"Andy", "Bart", "Carl", "Doug"}
```

(A) list.remove( list.size() );

(B) list.remove( 5 );

(C) list.clear("Elmo");

(D) list.remove( list.size()-1 );

(E) list.clear( );

21. The following is a piece of code that reads input from a file called `input.txt`. What is the format of each line of the input file that the code expects?

```
FileReader fr = new FileReader("input.txt");
BufferedReader br = new BufferedReader(fr);
String line = br.readLine();
int lineNum = 0;
while (line!=null) {
  String[] args = line.split("\t");
  lineNum++;
  int num = Integer.parseInt(args[1]);
  double other = Double.parseDouble(args[0]);
  line = br.readLine();
}
```

(A) Two `ints`, then a `double`, separated by tabs

(B) A `double`, a space character, then an `int`

(C) An `int`, a space character, then a `double`

(D) A `String[]`

(E) A `double`, a tab character, then an `int`

22. What does the following code print?

```
HashSet <Integer> appleGrades = new HashSet <Integer> ();
appleGrades.add (1);
appleGrades.add (2);
appleGrades.add (3);
appleGrades.add (1);
appleGrades.add (3);
System.outprintln(appleGrades.size());
```

(A) 4

(B) 3

(C) 6

(D) 5

(E) 10

23. What does the following code print?

```
double x = 2.0;
if (x > 1.0) {
  x /= 2;
}
if (x <= 1.0) {
  x /= 2;
}
System.out.println(x);
```

(A) 1

(B) 1.0

(C) 0.0

(D) 0

(E) 0.5

24. When an object no longer has any variables referring to it, what happens to it?

(A) It is sent to the Dumpster.

(B) The garbage collector can free the memory it occupied to make space for new data.

(C) Java refers to it using another variable.

(D) It is swapped out to disk.

(E) It sits around in main memory until you reboot your computer.

25. Consider the following class Person,

```
public class Person {
    private String name = "";
    public void setName( String name ) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}
```

What does the following code print?

```
public class Test {
    public static void main(String[] args) {
        Person p = new Person();
        p.setName("Alice");
        build(p);
        System.out.println(p.getName());
    }

    public static void build(Person p) {
        p.setName("Bobby");
        p = new Person();
        p.setName("Christy");
        p = new Person();
        p.setName("Derek");
    }
}
```

(A) Christy

(B) Bobby

(C) Alice

(D) This is a compile-time error.

(E) Derek

26. What does the following code print?

```
public class Operation{
  private int data=50;

  public void change(int data){
    data=data+100;
  }

  public static void main(String args[]){
    Operation op=new Operation();
    op.change(50);
    System.out.print(op.data+", ");
    op.change(500);
    System.out.println(op.data);
  }
}
```

(A) 100, 600

(B) 200, 800

(C) 150, 250

(D) 150, 150

(E) 50, 50

27. **This question and the next question will be used to implement the computation of a method called `sumSquares`, which has the following structure:**

```
public static int sumSquares(int n) {
  // Q27. base case
  // Q28. recursive step
}
```

This method expects a *non-negative* integer `n`, and returns the sum of all squares from 0 up to and including $n^2$. For example, `sumSquares(4)` returns $30 = 0^2 + 1^2 + 2^2 + 3^2 + 4^2$. How should the base case be implemented?

(A) `if (n == 0) return sumSquares(n - 1);`

(B) `if (n == 0) return 0;`

(C) `if (n == -1) return 1;`

(D) `if (n == 1) return sumSquares(n - 1);`

(E) `if (n == 1) return 1;`

28. Which of the following is a correct implementation of the recursive step?

(A) `return sumSquares(n) * sumSquares(n);`

(B) `return sumSquares(n - 1) + 2 * n - 1;`

(C) `return sumSquares(n);`

(D) `return sumSquares(n - 1) + n * n;`

(E) `return sumSquares(n - 1) * sumSquares(n - 1);`

29. What are the following code print when the program is run with no command line arguments (eg: in DrJava with the command `run MyClass`)?

```
public class MyClass {
  public static void main(String[] args) {
    try {
      System.out.println(foo(args));
    }
    catch (ArrayIndexOutOfBoundsException e2) {
      System.out.println("Ugh!");
    }
    catch (Exception e) {
      System.out.println("Gah!");
    }
  }

  public static String foo(String[] arr) {
    try {
      return arr[0];
    }
    catch (NullPointerException e) {
      System.out.println("Argh!");
    }
    return "No!";
  }
}
```

(A) `Argh!`
`No!`

(B) `Ugh!`

(C) `Argh!`

(D) `Gah!`

(E) Nothing. The program will crash.

## Multiple Answer Multiple Choice (2 points each)

In this section, the questions may have more than one correct answer. In order to get full marks, shade in *all* correct answers on your scantron sheet. No part marks will be given.

30. In which of the following does b have the value `true`?

    (A) boolean b = 0 == 3/4;

    (B) boolean b = 0.75 == (double)(3/4);

    (C) boolean b = 0.75 == (double) 3/4;

    (D) boolean b = 0 == -3/4;

    (E) boolean b = 0.5 == 1/2;

31. Which of the following will **not** compile? (Assume any required imports are written in the code).

    (A) Integer i = 3;

    (B) ArrayList<int> nums = new ArrayList<int>();

    (C) int x = Math.pow(2,3);

    (D) String s = null;

    (E) int[] nums = {1,2,3};
    int x = nums[3];

32. Which of the following are valid ways to *overload* a method whose header is:
    `public void method(int x)`?

    (A) `public void method(int x, int y)`

    (B) `public static void method(int x)`

    (C) `public void method(int z)`

    (D) `public String method(int x)`

    (E) `public void method1(int x)`

33. Which of the following initializations will result in the number 29.95 being printed to the console?

```
//variable initalization
System.out.println(d);
```

    (A) Double d = 29.95;

    (B) int d = 29.95;

    (C) Double d = new <Double>(29.95);

    (D) Double d = new ArrayList<Double>(29.95);

    (E) Double d = new Double(29.95);

34. Which of the following prints all of the elements in the array `int[] a = {1,2,3,4,5};`?

(A) System.out.println(a);

(B) System.out.print(a.values());

(C)
```
for(int x:a}
  System.out.println(x);
}
```

(D)
```
int x = 0;
while(x<a.length){
  x++;
  System.out.println(a[x]);
}
```

(E)
```
for(int i=0; i<a.length; i++){
  System.out.println(a[i]);
}
```

35. Recall the `Robot` class from Assignment 3. This class has a *non-static* method called `move` which moves a `Robot` object forwards one square. Suppose we wanted to write a method called `moveFar` that allows the robot to move forwards `n` squares. Which of the following are valid ways to write and call this method in the `RobotMoveLights` class? Assume that the Robot class has a constructor that takes no input.

(A)
```
public static void moveFar(Robot r, int n){
  for(int i=0; i<n; i++){
    r.move();
  }
}
//The lines below are in the main method
Robot robot = new Robot();
moveFar(robot, 5);
```

(B)
```
public static void moveFar(Robot r, int n){
  for(int i=0; i<n; i++){
    r.move();
  }
}
//The lines below are in the main method
Robot robot = new Robot();
robot.moveFar(robot, 5);
```

(C)
```
public static void moveFar(Robot r, int n){
  if(n>0){
    r.move();
    moveFar(r, n-1);
  }
}
//The lines below are in the main method
Robot robot = new Robot();
moveFar(robot, 5);
```

(D)
```
public static void moveFar(int n){
  if(n>0){
    Robot.move();
    Robot.moveFar(n-1);
  }
}
//The lines below are in the main method
Robot robot = new Robot();
robot.moveFar(5);
```

(E)
```
public static void moveFar(Robot r, int n){
  r.move();
  if(n>0){
    moveFar(n-1);
  }
}
//The lines below are in the main method
Robot robot = new Robot();
moveFar(robot, 5);
```

# Coding Section

Answer the following questions by writing your solutions in the provided booklet.

36. **Stopwatch**

For this question, you will define two new types. You will then write a program that uses those types.

Write a class `Time`. A `Time` object represents a time in a day by the 24-hour clock, and must consist of three private `int` attributes (a.k.a., fields, or properties): `hour`, `minute`, and `second`. `hour` must be between 0 and 23. `minute` and `second` must be between 0 and 59.

In addition to its properties, your `Time` class must have the following methods:

(A) A constructor that takes as input 3 ints and sets the private attributes appropriately.

(B) A method `getHour()` which returns the value stored in the `hour` attribute.

(C) A method `getMinute()` which returns the value stored in the `minute` attribute.

(D) A method `getSecond()` which returns the value stored in the `second` attribute.

Now that you have a new type, you will write a program to use it.

We will write a class called `Stopwatch`, which can be used to measure the progression of time. The `Stopwatch` class has a private boolean attribute called `running`, which represents whether or not the stopwatch is currently running and is false by default. It also has a private attribute of type `Time` called `startTime`, which represents the time at which the Stopwatch was started. You do not need to write a constructor.

(E) Write a method `startStop` that does one of the following, depending on whether the stopwatch is currently running. If the stopwatch is currently not running, it will start the running of the stopwatch and record the current time. If the stopwatch is currently running, it will stop the stopwatch, and print out how many seconds have elapsed since the stopwatch was started.

Assume that you have access to a method called `getCurrentTime()` (written by someone else inside of the Stopwatch class), which returns a `Time` object giving the current time to the nearest second. You may assume that the stopwatch does not run past midnight; i.e., both times will be in the same day.

(F) Next, write a main method in the Stopwatch class that tests the functioning of your code. Your main method must create an instance of the Stopwatch class when you begin running the program. It must then print a message prompting the user to "Enter any String to stop the stopwatch." The program uses `Scanner`'s `nextLine()` method to wait for a user response. When the user enters any response, the program will stop the Stopwatch and print out the number of seconds elapsed.

Here is a sample run of the final program:

```
> run Stopwatch
Enter any String to stop the stopwatch.
  /* Some time passes */
  COMP-202 is over. :-(    // Input by user
162123 seconds have elapsed.
>
```

Feel free to write any private helper methods you deem useful. This is not required.

SUMMARY OF JAVA STANDARD LIBRARY METHODS FOR SELECTED CLASSES

- `Arrays` (package `java.util.Arrays` Methods:
  - `public int[] copyOfRange(int[] original, int from, int to)`: Returns a subset of the `original` starting at `from` and finishing at `to`, excluding `to`. `to` might lie outside of the array.
- `String` (package `java.lang`) Methods:
  - `public boolean equals(Object anObject)`: Compares this `String` to `anObject`.
  - `public int length()`: Calculates the length of this `String`.
  - `public char charAt(int i)`: Gets the `char` at position `i` of the `String`. Note that counting starts from 0 so that to get the first character of the `String` you should input `i` equals 0.
  - `public boolean equalsIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString`.
  - `public int compareTo(String anotherString)`: Compares this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int compareToIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int indexOf(String str)`: Returns the index within this string of the first occurrence of the specified substring. If no such substring exists, returns -1.
  - `public int lastIndexOf(String str)`: Returns the index within this string of the last occurrence of the specified substring. If no such substring exists, returns -1.
  - `public String replace(char c, char d)` : Returns a new String with all occurrences of the character `c` in the `this` String replaced by the character `d`.
  - `public String replace(CharSequence target, CharSequence replacement)`: Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. (N.B.: A String is a kind of CharSequence.)
  - `public char[] toCharArray()`: Converts this `String` to a new character array.
  - `public String toLowerCase()`: Converts all of the characters in this String to lower case.
  - `public String[] split(String regex)`: Splits this string around matches of the given regular expression.
  - `public String substring(int start, int finish)`: Returns a new String composed of the `this` String starting from index `start` and up to, but not including index of `finish`
  - `public String toUpperCase()`: Converts all of the characters in this String to upper case.
- `File` (package `java.io`) Methods:
  - `public FileSstring pathname)`: Creates a new `File` instance that corresponds to the given pathname.
- `Scanner` (package `java.util`) Methods:
  - `public Scanner(Inputstream source)`: Constructs a new `Scanner` that produces values scanned from the specified input stream.
  - `public Scanner(File f)`: Constructs a new `Scanner` that produces values scanned from the specified File
  - `public double nextDouble()`: Scans the next token of the input as a `double`.
  - `public boolean nextBoolean()`: Scans the next token of the input as a `boolean`.
  - `public int nextInt()`: Scans the next token of the input as an `int`.
  - `public String nextLine()`: Advances this `Scanner` past the current line and returns the input read.
  - `public boolean hasNextLine()`: Checks whether there are further lines left to scan.
- `PrintStream` (package `java.io`) Methods:
  - `public void print(boolean b)`: Prints `boolean` value `b`.
  - `public void print(double d)`: Prints `double` value `d`.
  - `public void print(int i)`: Prints `int` value `i`.
  - `public void print(Object o)`: Prints `Object` `o`.
  - `public void print(String s)`: Prints `String` `s`.
  - `public void println()`: Terminates the current line by writing the line separator string.
  - `public void println(boolean b)`: Prints `boolean` value `b` and then terminates the line.
  - `public void println(double d)`: Prints `double` value `d` and then terminates the line.
  - `public void println(int i)`: Prints `int` value `i` and then terminates the line.
  - `public void println(Object o)`: Prints `Object` `o` and then terminates the line.
  - `public void println(String s)`: Prints `String` `s` and then terminates the line.
- `ArrayList` (package `java.util`) Methods:

  - `public boolean ArrayList<type>()`: Creates a new `ArrayList<type>`
  - `public void add(type t)` : Appends the specified element to the end of this list.
  - `public void add(int index, type t)`: Inserts the specified element at the specified position in this list.
  - `public void addAll(Collection<type> c)`: Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
  - `public boolean contains(Object o)`: Returns true if this list contains the specified element.
  - `public type get(int index)`: Returns the element at the specified position in this list.
  - `public int indexOf(Object o)`: Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. It searches for the object by calling the .equals() method defined on the Object.
  - `public boolean remove(Object o)`: Removes the first occurrence of the specified element from this list, if it is present.
  - `public int size()`: Returns the number of elements in this list.

- `Math` (package `java.lang`) Methods:

  - `public static double pow(double a, double b)`: Returns the value of `a` raised to the power of `b`.
  - `public static double sqrt(double a)`: Returns the correctly rounded positive square root of `double` value `a`.
  - `public static double random()`: Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
  - `public static double exp(double a)`: Returns Euler's number $e$ raised to the power of `double` value `a`. (base $e$) of `double` value `a`. of `double` value `a`.

```
public class Penguin{
  private String species;
  private String name;
  private double height; //height is in cm
  private boolean isAdult;

  public Penguin(String name, String species){
    this.name = name;
    this.species = species;
    this.isAdult = false;
    setHeightAtBirth(species);
  }
  private void setHeightAtBirth(String species){
    //little blue, or fairy penguins, are the smallest in the world!
    if(species.equals("little") || species.equals("fairy"))
         this.height = 7;
    if(species.equals("emperor") || species.equals("king"))
         this.height = 10;
    else
      this.height = 8;
  }
  public void growUp(){
    this.isAdult = true;
    if(species.equals("little") || species.equals("fairy"))
         this.height = 33;
    if(species.equals("emperor") || species.equals("king"))
         this.height = 100;
    else
      this.height = 60;
  }
  public static Penguin breed(Penguin mom, Penguin dad){
    Penguin p;
    if(!mom.isAdult || !dad.isAdult)
      return null;
    if(mom.species.equals(dad.species))
      p = new Penguin("baby", mom.species);
    else
      p = new Penguin("baby", "Mix");
    return p;
  }
  // Your toString() method goes here.
}
```