Lecture April 11 - Final Review

Bentley James Oakes

April 11, 2018

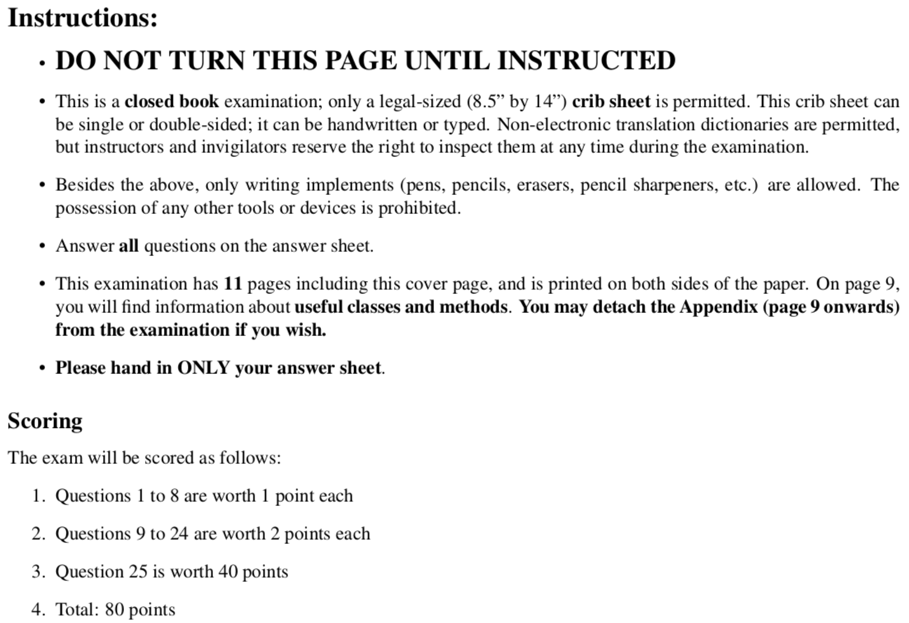
Schedule

- This lecture will discuss the final and some example questions
  - Tutorial dates/office hours will be on myCourses
- Friday (for MWF class) I will show the Final 2016 long answer and a case study
- Monday I will discuss future steps for programming and other programming languages
  - Non-testable material, but still relevant to programming

Final

- Final on **April 25th, 09:00 to 12:00**
- In the main gym, rows 2-40
- If you missed the midterm, your final exam will be worth 65% of your final grade

Final Front Page



Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** examination; only a legal-sized (8.5" by 14") **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** questions on the answer sheet.
- This examination has **11** pages including this cover page, and is printed on both sides of the paper. On page 9, you will find information about **useful classes and methods**. **You may detach the Appendix (page 9 onwards) from the examination if you wish.**
- **Please hand in ONLY your answer sheet**.

Scoring

The exam will be scored as follows:

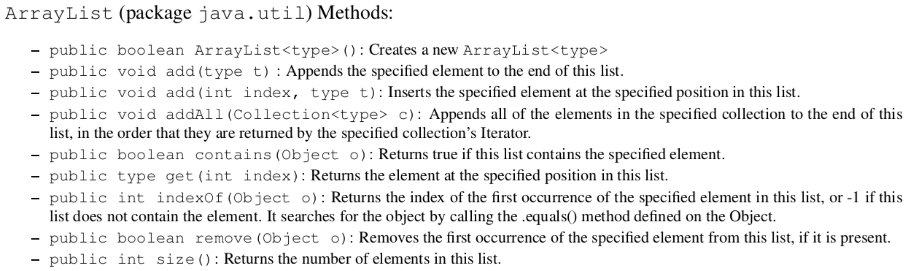
1. Questions 1 to 8 are worth 1 point each
2. Questions 9 to 24 are worth 2 points each
3. Question 25 is worth 40 points
4. Total: 80 points

Final

- (Same as the midterm)
- Closed book examination
- A legal-sized (8.5" by 14") crib sheet is permitted
  - Single or double-sided, handwritten or typed
- Non-electronic translation dictionaries are permitted
- Otherwise, only writing implements (pens, **pencils**, erasers, pencil sharpeners, etc.) are allowed
- Possession of any other tools or devices is prohibited.
- **Bring McGill ID** - Very important!

Final

- Documentation of `String`, `print`, `Math`, and `ArrayList` classes are provided



`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.

Notes

- Compared to the midterm, the final will focus more on reading code and figuring out what it does
  - Example: In the code provided at the end of the exam, how can this method be called?
- Sample finals (and some answers) will be posted on myCourses

Section 1

Material

Material

The final also covers material from the first half of the course:

- Binary
- Types, expressions, scope
- Errors and Exceptions
- Booleans, conditions, loops
- Methods
- `String` and `char`
- Arrays and reference types

Second Half - A

- `null`
  - What is null used for?
  - How can you check for null?
- Instantiating and using objects
  - Calling static and non-static methods of a class
  - Creating instances of a class
- Constructors
  - What's the header of a constructor?
  - What does a constructor do?
  - What is the purpose of the `this` keyword?

Second Half - B

- Overloading methods
  - When can we have methods with the same name?
- Access modifiers (`public`, `private`, `final`)
  - `private` means the attribute/method can't be accessed outside of the class
  - `public` is the opposite of `private`
  - `final` means that the attribute's value can't be changed once it is set
- Getters and setters
  - Why do we use getters and setters?
  - Can you write a getter and a setter for the `String name` attribute?

Second Half - C

- Immutable reference types
  - A class with private attributes and no setters
  - The value of an instance can't change once created
  - Copying `ArrayLists` and arrays, so that other classes don't get access to the data
- Static attributes and methods
  - `Static` means the attribute or the method belongs to the class
  - How can we access static attributes and methods in other classes?

Second Half - D

- Exceptions
  - When are `Exceptions` thrown?
    - `NullPointerException`
    - `ArrayIndexOutOfBoundsException`
  - How to `throw Exceptions`
  - How to properly `catch Exceptions`, and catch multiple `Exceptions`
  - The `try, catch, finally` blocks
  - Checked vs unchecked
  - How to mark a method with the `throws` keyword, and handle the Exception in another method

Second Half - E

- Using libraries (`Math`, `Random`, `Scanner`)
  - How to use these libraries
  - Put code for using `Scanner` on your cheat-sheet
- Wrapper classes
  - Like `Double`
  - Why do we need wrapper classes?
  - What is auto-boxing and auto-unboxing?
- `ArrayList` class
  - What types can an `ArrayList` store? (only reference types)
  - How do we use the `add` and `get` methods?

Not on the Final

What's **not** on the final?

- HashMaps
- Image editing
- Recursion
- Discussion of inheritance
  - Other than: all classes have a default constructor and `toString` method

How To Study

- Review slides, assignments
- Create your own crib sheet
- Ask questions at office hours, tutorials
- Watch myCourses for office hour/tutorial times
- Do past final exams on myCourses

- Review early and often
- Don't cram the night before

Section 2

Example Questions

Example 1

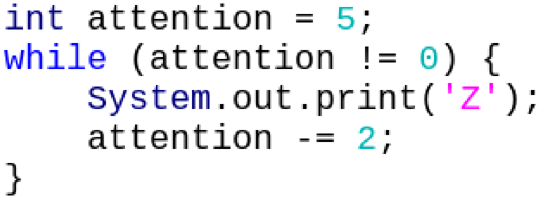
How many times does the following code print the character 'Z'?

```
int attention = 5;
while (attention != 0) {
    System.out.print('Z');
    attention -= 2;
}
```

- 3
- 4
- An infinite number of times. ZZZZZZZZZZZZZZZZZZZZ...
- 5
- 0

Example 1 - Answer

How many times does the following code print the character ‘Z’?



```
int attention = 5;
while (attention != 0) {
    System.out.print('Z');
    attention -= 2;
}
```

- Answer: An infinite number of times. ZZZZZZZZZZZZZZZZZZ...
- Why? `attention` is 1, then -1, then -3 and will never be 0

Example 2

```
import java.util.ArrayList;
public class BoxOfChocolates {
    public int chocolates;

    public BoxOfChocolates(int number) {
        chocolates = number;
    }

    public static void eat(ArrayList<BoxOfChocolates> boxes) {
        for (int i = 0; i < boxes.size(); i++) {
            if (boxes.get(i).chocolates > 0)
                boxes.get(i).chocolates--;
        }
    }

    public static void main(String[] args) {
        ArrayList<BoxOfChocolates> boxes = new ArrayList<BoxOfChocolates>();
        BoxOfChocolates myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        boxes.set(1, myBox);
        eat(boxes);
        for (int i = 0; i < boxes.size(); i++)
            System.out.print(boxes.get(i).chocolates + " ");
    }
}
```

Choices:

1. 2 1 1
2. 0 0 0
3. 2 2 2
4. 3 3 3
5. 2 1 0

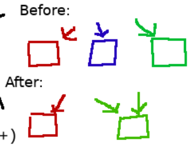
Example 2 Answer

```
import java.util.ArrayList;
public class BoxOfChocolates {
    public int chocolates;

    public BoxOfChocolates(int number) {
        chocolates = number;
    }

    public static void eat(ArrayList<BoxOfChocolates> boxes) {
        for (int i = 0; i < boxes.size(); i++) {
            if (boxes.get(i).chocolates > 0)
                boxes.get(i).chocolates--;
        }
    }

    public static void main(String[] args) {
        ArrayList<BoxOfChocolates> boxes = new ArrayList<BoxOfChocolates>();
        BoxOfChocolates myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        myBox = new BoxOfChocolates(3);
        boxes.add(myBox);
        boxes.set(1, myBox);
        eat(boxes);
        for (int i = 0; i < boxes.size(); i++)
            System.out.print(boxes.get(i).chocolates + " ");
    }
}
```



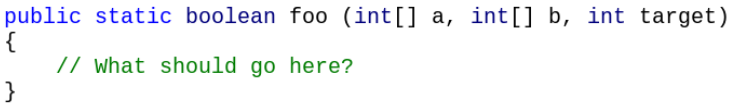
Answer: 2 1 1

Example 3

The following method takes two integer arrays and a target integer as input. Returns `true` if there is an element from the first array and an element from the second array whose sum is the target integer.

For instance, given arrays `a=[1,4,3]` and `b=[2,6,3]` and target `7`, the method should return `true` because `a[1]+b[2] = 4+3 = 7`. However, if the target is `8`, it should return false, as no element in array `a`, added with an element of array `b` results in `8`.

What should be the body of this method?



```java
public static boolean foo (int[] a, int[] b, int target)
{
    // What should go here?
}
```

Example 3 Options

(A)
```
for(int j = 0; j < b.length; j++)
{
    for(int i = 0; i < a.length; i++)
    {
      if(a[i] + b[j] == target)
        return true;
    }
}
return false;
```

(B)
```
for(int i = 0; i < a.length + b.length; i++)
{
    if(a[i] + b[i] == target)
      return true;
    else
      return false;
}
```

(C)
```
for(int i = 0; i < a.length; i++)
{
    if(a[i] + b[i] == target)
      return true;
    else
      return false;
}
```

(D)
```
for(int i = 0; i < a.length + b.length; i++)
{
    if(a[i] + b[i] == target)
      return true;
}
return false;
```

(E)
```
for(int i = 0; i < a.length; i++)
{
    for(int j = 0; j < b.length; j++)
    {
      if(a[i] + b[j] == target)
        return true;
      else
        return false;
    }
}
```

Takes two integer arrays and a target integer as input. Returns `true` if element from the first array plus element from the second array equals the target integer.

Answer: **A**

Example 4

Assume a `Penguin` class with a name, breed, and height

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() { return name + " " + breed + " " + height; }`

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

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

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

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

- Answer: A

Example 5

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) 40.0

(B) A string “40”

(C) There is a compile-time error.

(D) 20

(E) 40

- Answer: C
- Why? Because bottom two method headers are the same

Example 6

What does the following code print?

```java
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) 150, 250

(C) 200, 800

(D) 50, 50

(E) 150, 150

Answer: D

Because the `this` keyword is missing from the `change` method

Example 7

True or false: The following example of a Dog class will compile without error.

```
public class Dog {
    public Dog(String name, int age){
        System.out.println("hi!");
    }
    public Dog2(String color) {
        System.out.println(color);
    }
    public static void main(String[] args){
        Dog dog1 = new Dog("Jack",5);
        Dog dog2 = new Dog2("Jill");
    }
}
```

- Answer: False
- Why? Because the second constructor must be called Dog

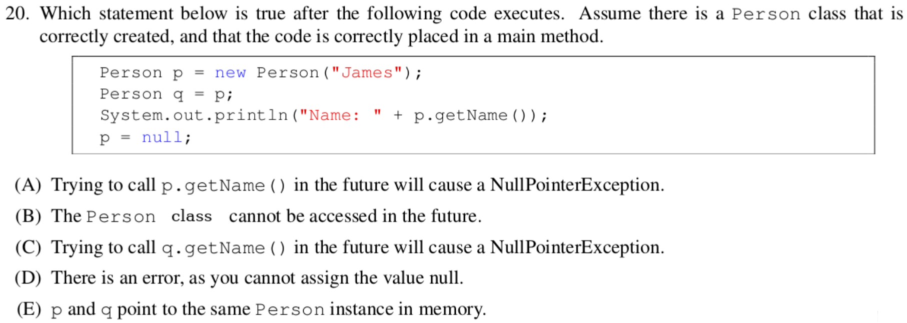
Example 8

What gets printed when the following program is run?

```
public class Test2 {
    public static void main(String[] args) {
        int number = 11;
        boolean answer = true;
        for (int i = 2; (i < number) && answer ; i++){
            if (mysteryOperation(number, i)){
                answer = false;
            }
        }
        System.out.println(answer);
    }
    public static boolean mysteryOperation(int x, int y) {
        return ((x % y)==0);
    }
}
```

- Answer: *true*
- Explanation: *i* ranges from 0 to 10
- The method asks whether 11 mod *i* is equal to 0
- In other words, is 11 prime? Yes, so answer remains as *true*

Example 9



20. Which statement below is true after the following code executes. Assume there is a `Person` class that is correctly created, and that the code is correctly placed in a main method.

```
Person p = new Person("James");
Person q = p;
System.out.println("Name: " + p.getName());
p = null;
```

(A) Trying to call `p.getName()` in the future will cause a NullPointerException.

(B) The `Person` class cannot be accessed in the future.

(C) Trying to call `q.getName()` in the future will cause a NullPointerException.

(D) There is an error, as you cannot assign the value null.

(E) `p` and `q` point to the same `Person` instance in memory.

- Answer: *A*

Example 10a

For the next questions, assume there exists an `Elephant` class with two non-static attributes:

`int age` and `String name`

1. You are writing a *static* method called `getAge` inside of the `Elephant` class that returns the value of the given `Elephant's` age attribute. Write just the method header. Do not write the entire method.
   - `public static int getAge(Elephant e)`
2. Assume you have access to an `Elephant` instance `e1`. Call the `getAge` method to obtain the age of the elephant `e1` and store it in a variable. Write one statement to do this. Assume that you are writing code for outside of the `Elephant` class.
   - `int age = Elephant.getAge(e1);`

Example 10b

For the next questions, assume there exists an `Elephant` class with two non-static attributes:

`int age` and `String name`

1. You are writing a *non-static* method called `setName` inside of an `Elephant` class that sets the value of the `name` attribute. Write just the method header. Do not write the entire method.
   - `public void setName(String name)`
2. Assume you have access to an `Elephant` instance `e1`. Call the `setName` method to assign the name of `Dumbo` to the elephant `e1`. Write one statement to do this. Assume that you are writing code for outside of the `Elephant` class.
   - `e1.setName("Dumbo");`

Example 11

```java
public class FinalExam {
    public static void main(String[] args) {
        try {
            func();
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("main");
        }
    }
    public static void func() {
        try {
            String foo = null;
            System.out.println(foo.length());
        }
        catch (NullPointerException e) {
            int[] bar = {4,5,6};
            System.out.println(bar[100]);
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("func");
        }
    }
}
```

What is printed by this code?

- Answer: *main*
- Inside method, `NullPointerException` is thrown and caught
- Inside that catch block, `AIOOBException` is thrown, and caught in main method
- The second catch block can only catch `AIOOBExceptions` within the attached `try` block

Section 3

Wrapup

Course Summary

What have we done?You learned how to:

- Break down problems into small pieces
- Tie together different methods and classes
- Follow the flow of information and control
- Write instructions to a computer

Course Evaluations

- Course evaluations are now up on MyCourses
- Please let me know honest feedback
- I read them all, and the feedback lets me know how to improve