

Mitderm I Review

Material borrowed from PNH, JRS, Nam Mai, Erin Korber

1 Quick Execution

What is the result of executing the following code:

1. Execution 1

```
public class IntList{
    int head;
    IntList tail;
    public IntList(int num) {
        head = num;
        tail = null;
    }
    public String toString() {
        String s = "[ " + head;
        IntList temp = tail;
        while (temp != null) {
            s = s + ", "+ temp.head;
            temp = temp.tail;
        }
        s = s+ " ]";
        return s;
    }
}
```

In some other class:

```
public void create5(IntList l) {
    l = new IntList(5);
}
public static void main(String[] args) {
    IntList list = new IntList(4);
    create5(list);
    System.out.println(list);
}
```

Output: Compile-time Error, Run-time Error, Output(please specify)

2. Execution 2

```
int i = 1;
int j = i;
i = 2;
System.out.println(j);
```

Output: Compile-time Error, Run-time Error, Output(please specify)

3. Execution 3

```
System.out.println("Perfect Quiz:");
System.out.println(7+8+10);
System.out.println("Realistically: "+6+1+0);
System.out.println(6+2+3 + "In the end");
```

Output: Compile-time Error, Run-time Error, Output(please specify)

2 Q & A

Answer the following questions as concisely as possible.

1. Please draw the Box and Pointer diagram

```
String[] sa = new String[3];
sa[0] = "lab1";
sa[1] = sa[0];
sa[2] = sa[1].substring(0, 3);
```

2. When would you make a method static instead of non-static?

3. Why would you declare variables as protected instead of private?

4. What class has no superclass?

5. Why are most final fields also static?

6. Object o = s.substring(0); Fill in the below <>

The above line causes a <compile time/run-time> error if <condition>

The above line causes a <compile time/run-time> error if <condition>

3 Inheritance of Knowledge

Framework Code

Suppose we had the following classes:

```

abstract class AbstractBook {
    String title;
    String author;
    int totalPages;
    public abstract void read(int num);
    public static void summary (Book b) {
        System.out.println(b.title + " written by " + b.author);
    }
}

interface Comparable {
    public int compareTo(Object obj);
}

class Book extends AbstractBook implements Comparable {
    public Book (String title, String author, int totalPages) {
        this.title = title;
        this.author = author;
        this.totalPages = totalPages;
    }
    /* compareTo will be used to catalog are books. The ordering is
    * first by Author and then by Title. You can use Strings
    * compareTo function to help.
    */
    public int compareTo(Object obj) {
        /* Please fill this in */
        return 0;
    }
    public void read (int num) {
        System.out.printf("You read page %d.\n", num);
    }
    public void work (int num) {
        System.out.printf("You work on page %d.\n", num);
    }
}

class TextBook extends Book {
    public TextBook (String title, String author, int totalPages) {
        super (title, author, totalPages);
    }
    public void read (int num) {
        System.out.printf("You read page %d of %s closely and learn something new.", num, title);
    }
    public void work (int num) {
        System.out.printf("You do the exercises on page %d.", num);
    }
    public void compareTo (TextBook book) {

```

```
        if (this.totalPages <= book.totalPages)
            System.out.printf("%s isnt bigger than %s.", this.title, book.title);
        else
            System.out.printf("%s is bigger than %s.", this.title, book.title);
    }
}
class CSBook extends TextBook {
    public CSBook (String title, String author, int totalPages) {
        super(title, author, totalPages);
    }
    public void work (int num) {
        System.out.printf("You code up the exercise on page %d according to the specs.", num);
    }
}
class MathBook extends TextBook {
    public MathBook (String title, String author, int totalPages) {
        super(title, author, totalPages);
    }
    public void work (int num) {
        System.out.printf("You cant figure out how to the problems on page %d.", num);
    }
    public void compareTo (MathBook book) {
        if (this.totalPages <= book.totalPages)
            System.out.printf("%s isnt better than %s.", this.title, book.title);
        else
            System.out.printf("%s has more problems than %s.", this.title, book.title);
    }
}
class ColorBook extends Book {
    public ColorBook (String title, String author, int totalPages) {
        super(title, author, totalPages);
    }
    public void work (int num) {
        System.out.printf("You color in the figures on page %d.", num);
    }
}
```

Code Segments

Treat each segment of code independently, and describe what would happen in each case.

```
public class BookTest{
    public static void main (String[] args) {
        AbstractBook abst;
        Book book;
        CSBook cs;
        MathBook math;
        TextBook txt;
        ColorBook color;
        /* Consider the following blocks of code independently */
        /* --- 1 --- */
        abst = new AbstractBook ();
        abst.title = "The Myth of the Cave";
        abst.author = "Plato";
        abst.totalPages = 40;
        AbstractBook.summary(abst);
        /* --- 2 --- */
        book = new ColorBook("Doodles", "Smith", 100);
        AbstractBook.summary(book);
        book.read(20);
        /* --- 3 --- */
        book = new ColorBook("Doodles", "Smith", 100);
        book.work(45);
        /* --- 4 --- */
        book = new ColorBook("Doodles", "Smith", 100);
        ((ColorBook) book).work(45);
        ((TextBook) book).work(45);
        /* --- 5 --- */
        txt = new CSBook ("SICP", "Abelson", 750);
        math = new MathBook ("Linear Algebra", "Knutson", 400);
        MathBook math2 = new MathBook("Differential Equations", "Knutson", 420);
        math.compareTo (txt);
        math.compareTo (math2);
        txt = math2;
        math.compareTo (txt);
        /* --- 6 --- */
        abst = new CSBook ("GamesCrafters", "Garcia", 800);
        ((TextBook) abst).work(200);
        txt = new TextBook ("Game Theory", "Nash", 1300);
        ((CSBook) abst).compareTo(txt);
    }
}
```

Result of Execution:

- 1.
- 2.
- 3.
- 4.
- 5.

6.

4 Code Away

Given the information, please complete the reverse methods:

```
public class SListNode {
    Object item;
    SListNode next;
    public SListNode(Object item) {
        this.item = item;
        next = null;
    }
}
```

Suppose the following methods are in the SListNode Class:

1. Reversing a List

```
/** return the reverse of L (non-destructively). */
static SListNode reverse (SListNode L) {
    /* your answer */
}
```

2. Destructively Reversing a List

```
/** return the reverse of L (destructively).
 * (ie. mutate the old list and return the new
 * head of the list)
 */
static SListNode reverse2 (SListNode L) {
    /* your answer */
}
```

3. Inversion on a List

```
/** Does an inversion on L (destructively).
 * Puts the items furthest from the center of the list
 * to the center of the list. The centermost item (if L is odd)
 * is unaffected.
 * ie. List{1 2 3 4 5 6 7 8} inversions to {4 3 2 1 8 7 6 5}
 *    List{1 2 3 4 5 6 7 8 9} inversions to {4 3 2 1 5 9 8 7 6}
 */
```

```
static SListNode inversion (SListNode L) {  
    /* your answer */  
}
```