HKN CS61B Midterm 1 Review

Anthony Sutardja
Harry Wallace
Riyaz Faizullabhoy

Stack & Heap

```
class Ref1{
    public static void main(String[] args){
        int lol = 5;
        int wat = 10;
        lol = wat;
        lol = lol + 1;

        System.out.println(wat);
    }
}
```

What does the `main` method print out?

10

Stack & Heap

```
class Ref2{
    public static void main(String[] args){
        String a = "Cookies";
        String b = "Pizza rolls";
        String c = a;
        a = b;

        System.out.println(c);
    }
}
```

What does the `main` method print out?

Cookies

Stack & Heap

```
Burrito yum = new Burrito();
Food f = yum;
boolean check = (f == yum);
System.out.println(check);
```

What does the following print?

true

Stack & Heap

```
class Stack{
    public static void change1(int[] i){
        change2(i);
        i[0] = i[0] + 100;
    }

    public static void change2(int[] i){
        int[] j = {0, 0, 0};
        i = j;
    }

    public static void main(String[] args){
        int[] arr = {1, 2, 3};
        change1(arr);
        System.out.println(arr[0]);
    }
}
```

What does the `main` method print out?

101

Doubly Linked Lists!

```
public class DList {
 protected DListNode head;
 protected DListNode tail;

  public void moveToEnd() {
    // Moves the first node of the list to the end
    /* YOUR CODE HERE */
  }
}
```

Fill in `moveToEnd()`, a method to move the head of the DList to its tail.

Assume you have access to `next` and `prev` attributes.

Doubly Linked Lists!

```
public class DList {
 protected DListNode head;
 protected DListNode tail;

  public void moveToEnd() {
    // Moves the first node of the list to the end
    if(head != null) {
      tail.next = head;
      head.prev = tail;
      tail = head;
      head = head.next;
      tail.next = null;
      head.prev = null;
    }
  }
}
```

Try drawing the pointers!

Singly Linked Lists!

```
public void trim(int min, int max, SList list) {
  /* YOUR CODE HERE */
}
```

Write a method `trim` that will **destructively** remove elements from the input singly-linked list which have a value less than min or greater than max.

You can assume the input list is not null, and is comprised of `SListNode`'s

You can also assume the `head` attribute in `SList`, and the `next` and `value` attributes in `SListNode`

Singly Linked Lists!

```
public void trim(int min, int max, SList list) {
  SListNode curr = list.head.next;
  SListNode prev = list.head;
  while (curr != null) {
    if ((curr.value < min) || (curr.value > max)) {
      prev.next = curr.next;
      curr = curr.next;
    } else {
      prev = curr;
      curr = curr.next;
    }
  }
  if ((list.head.value < min) || (list.head.value > max)) {
    list.head = list.head.next;
  }
}
```

Try drawing the pointers!

Singly Linked Lists!

Write a method to return a **new** singly-linked list that only contains the odd elements of the input list.

```
public SList odd_list(SList list) {
  /* YOUR CODE HERE */
}
```

You can assume the input list is not null, and is comprised of `SListNode`'s

You can also assume the `head` attribute in `SList`, and the `next` and `value` attributes in `SListNode`

Singly Linked Lists!

```
public SList odd_list(SList list) {
  SList oddList = new SList();
  boolean addedHead = false;
  SListNode curr = list.head;
  SListNode newHead = newSListNode();
  SListNode prev;
  while (curr != null) {
    if (curr.value % 2 == 1) {
       if (!addedHead) {
           newHead.value = curr.value;
           oddList.head = newHead;
           prev = newHead;
           addedHead = true;
       } else {
           SListNode oddNode = new SListNode();
           oddNode.value = curr.value;
           prev.next = oddNode;
           prev = oddNode;
       }
    }
    curr = curr.next;
  }
  return oddList;
}
```

Inheritance

Basics:

- Natural class hierarchy!
- Subclasses **extend** Superclasses

```
A a = new B();
```

Static Type

Dynamic Type

Inheritance

Constructors:

There is **always** an implied call to the superclass constructor on the **FIRST LINE.**

```
class Child extends Parent {
    public Child() {
        System.out.println("hi!");
    }
}
```

```
class Child extends Parent {
    public Child() {
        super();
        System.out.println("hi!");
    }
}
```

these are the same!

(side note: the explicit call to super() can only be in the first line)

Inheritance

Dynamic Method Lookup:

If we **override** a method in a subclass, it will always be called if the dynamic type of our object is that of the subclass

overridden methods must have the same **signature** (name, arguments)

Field Shadowing:

We always consider the **static type** for looking up attributes
(for example, object.attribute will look at the object's static type)

Inheritance

Putting it all together:

Making an object:

1. If the static type is a subclass of the dynamic type, COMPILE-TIME ERROR

    ex: Cat c = new Animal();

2. Call the superclass constructor executes **first**, then execute the subclass constructor (implicit call to super();)

Casting:

1. We can cast "up" to any superclass without any problems

    ex: Cat c = new Cat();

        ((Animal) c) is a valid cast.

2. We can only cast "down" to the object's dynamic type

    --> if we cast "down" to a subclass below the dynamic type, RUN-TIME ERROR

    ex: Animal a = new Animal();

        ((Cat) a) will give us a run-time error!!

Inheritance

Putting it all together:

Calling Methods

1. Does the static type of the object have this method?

   --> If not, COMPILE TIME ERROR

   ex: Cat c = new Cat();

   c.woof(); would give us a compile time error, assuming Cat/its superclass doesn't define the method woof

2. Now, look at the dynamic type of the object -- is this method overridden?

   --> If the signature is the **EXACT SAME**, **use this method!** This is dynamic method lookup.

Attribute Lookup

1. Does the static type of the object have this attribute?

   --> If so, **use this value!** If not, COMPILE TIME ERROR

(inspiration from Summer '12, MT1)

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Superhero();
superhero.punch(superhero);
```

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Superhero();
superhero.punch(superhero);
```

```
"I'M A SUPERHERO"
"BOOM I'M A SUPERHERO"
```

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Batman batman = new Batman("I'M
BATMAN!");
batman.punch(batman);
```

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Batman batman = new Batman("I'M BATMAN!");
batman.punch(batman);
```

"I'M A SUPERHERO"
"I'M BATMAN!"
"wat."

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Batman batman = new Superhero();
batman.punch(batman);
```

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Batman batman = new Superhero();
batman.punch(batman);
```

COMPILE-TIME ERROR!

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Batman();
superhero.punch( (Batman) superhero);
```

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Superhero superhero = new Batman();
superhero.punch( (Batman) superhero);
```

```
"I'M A SUPERHERO"
"BOOM I'M A SUPERHERO"
"BOOM BATMAN!"
```

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Batman batman = new Batman();
((Superhero) batman).punch(batman);
```

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Batman batman = new Batman();
((Superhero) batman).punch(batman);
```

"I'M A SUPERHERO"
"BOOM I'M A SUPERHERO"
"BOOM BATMAN!"

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Superhero();
superhero.punch( (Batman) superhero);
```

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Superhero();
superhero.punch( (Batman) superhero);
```

RUN-TIME ERROR!

Inheritance

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Superhero superhero = new Batman();
superhero.punch( (Batman) superhero);
```

Inheritance

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }

    public void punch(Superhero a) {
        System.out.println("BOOM " + s);
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Superhero superhero = new Batman();
superhero.punch( (Batman) superhero);
```

"I'M A SUPERHERO"
"BOOM I'M A SUPERHERO"
"BOOM BATMAN!"

Inheritance

NOTE the changed source code!

```
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }
}
```

```
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```
Batman batman = new Batman();
((Superhero) batman).punch(batman);
```

Inheritance

NOTE the changed source code!

```java
public class Superhero {
    String s;
    public Superhero() {
        s = "I'M A SUPERHERO";
        System.out.println(s);
    }

    public void punch() {
        System.out.println("Punch! Punch!");
    }
}
```

```java
public class Batman extends Superhero {
    String s;
    public Batman() {
        s = "NANANANANANANA";
    }

    public Batman(String s) {
        this.s = s;
        System.out.println(this.s);
    }

    public void punch(Superhero v) {
        s = "BATMAN!";
        super.punch(v);
        System.out.println("BOOM " + s);
    }

    public void punch(Batman b) {
        System.out.println("Wat.");
    }
}
```

```java
Batman batman = new Batman();
((Superhero) batman).punch(batman);
```

COMPILE-TIME ERROR

Good luck on your midterm!

HKN Office Hours

Monday - Friday, 11:00am-5:00pm
345 Soda
290 Cory

hkn.eecs.berkeley.edu