CS2030S Recitation

Week 8: Problem Set 5

brian

2025-10-08

National University of Singapore

Recap

Recap: Stack and heap

- Recall that the Stack contains frames for the active method calls
- heap contains the objects in memory
- There's a 3rd space called the *metaspace*
  - stores class level information
  - static fields

Recap: Variable capture

- variables will disappear when the stack frame is popped
- What if an object requires the value of that variable?
  - Capture the variable in the object instance
  - Something like a “hidden” field (not accessible by programmer)

Recap: Immutability

- Setting the stage for another paradigm
- A DS is mutable if it can be changed
- Immutable means that it cannot be changed
  - `String` is immutable
- In this course we relax this definition
  - Immutable objects must not have *observable* changes on the *outside*

Recap: Why make things immutable

- Easier to reason about
  - Guarantees that whatever you are referring to has not changed
- Sharing objects
  - Multiple objects can refer to something without worry about it changing
- Sharing internals
  - Possible to reuse some internals (see notes example on `ImmutableSeq`)
- Safer concurrency
  - Guarantees would still hold even if different interleaving of instructions (not important now, will learn in the future)

Recap: Fully qualified name

- Consider the following code

```
class B{
  int x = 1;

  class A {
    int x = 0;

    int f() {
      int x = 3;
      return x; // which x?
    }
  }
}
```

- What is `x` referring to?
- Somewhat ambiguous, esp from the compilers pov
- Fully qualify the name to remove ambiguity

Recap: Fully qualified name

- Consider the following code

```java
class B{
  int x = 1;

  class A {
    int x = 0;

    int f() {
      int x = 3;
      return B.this.x; // oh this x
    }
  }
}
```

- What is x referring to?
- Somewhat ambiguous, esp from the compilers pov
- We can fully qualify the name to remove ambiguity

Recap: Fully qualified name

- Idea:
  - If it's a field add `this`
  - If it's some outer class, add the class name e.g. `B`
  - we can chain these 2 e.g. `B.this` to access outer class `B`'s fields

Q1:

We have the following code

```
B b = new B();
B.f();
```

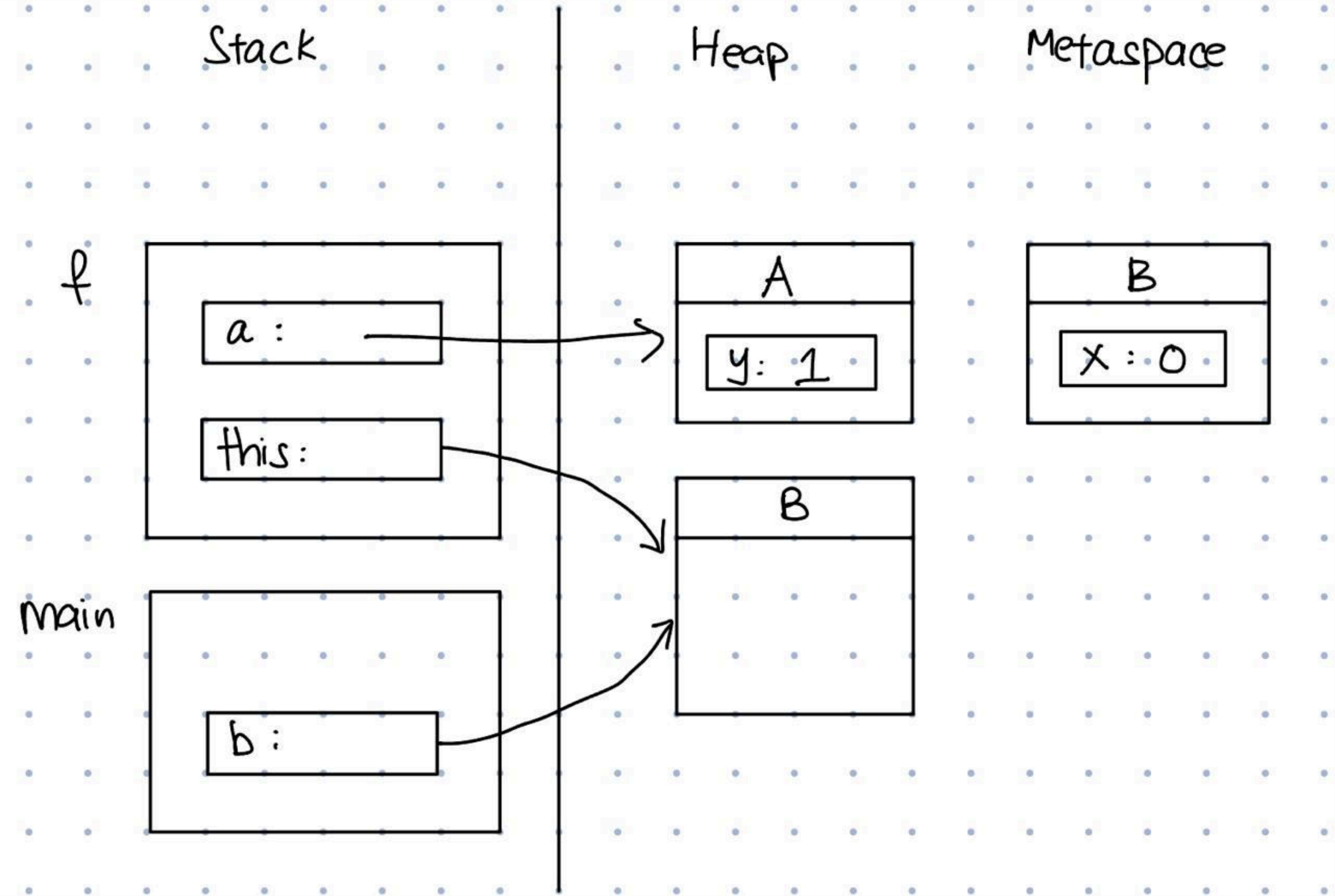
Q1a: Fully qualify and draw stack and heap

```java
class B {
  static int x = 0;
  void f() {
    A a = new A();
    // Line A
  }
  static class A {
    int y = 0;
    A () {
       y = x + 1;
    }
  }
}
```

Q1a: Fully qualify and draw stack and heap

```
class B {
  static int x = 0;
  void f() {
    A a = new A();
    // Line A
  }
  static class A {
    int y = 0;
    A () {
      this.y = B.x + 1;
    }
  }
}
```

Q1a: Fully qualify and draw stack and heap



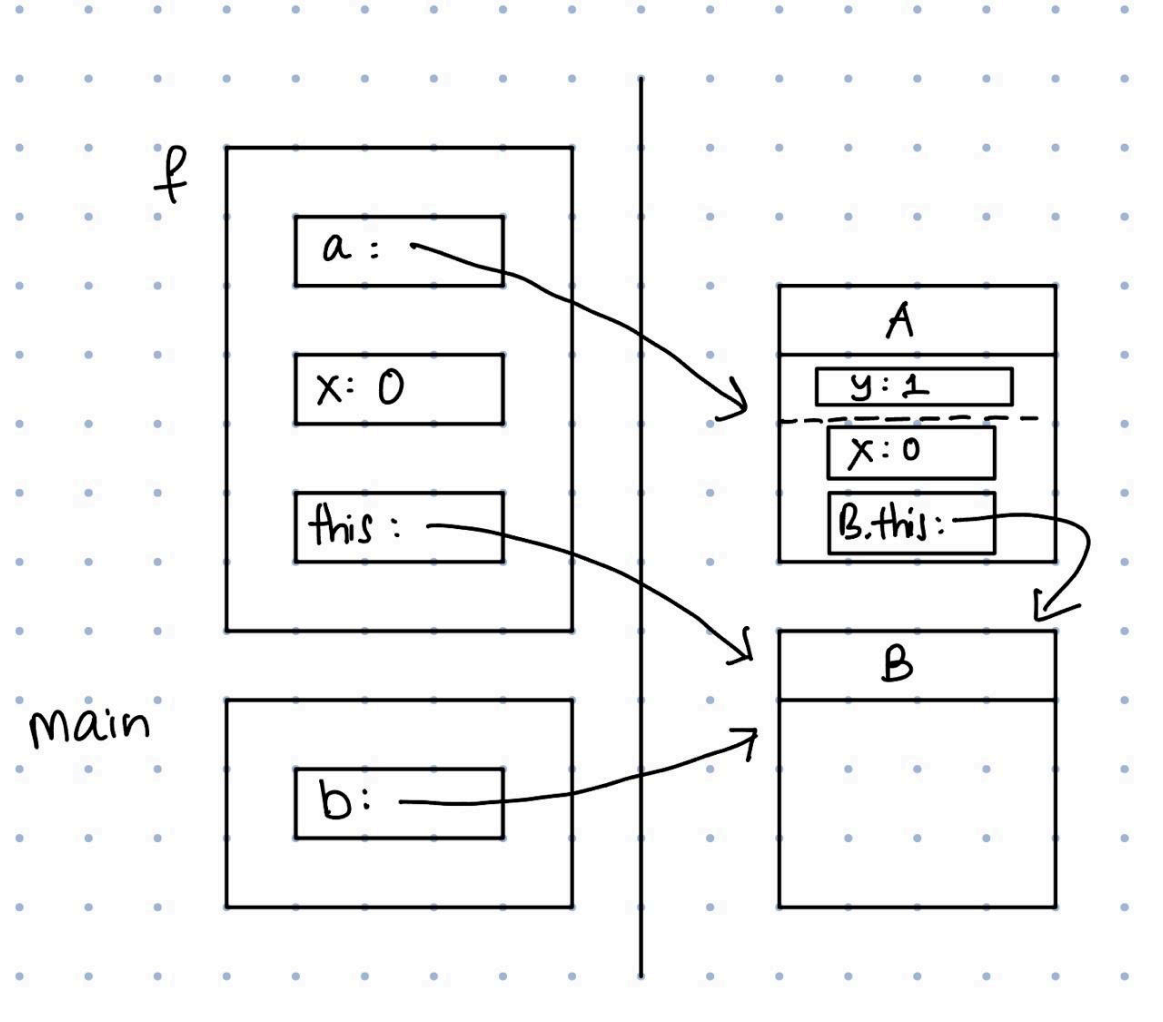
Q1b: Fully qualify and draw stack and heap

```java
class B {
  void f() {
    int x = 0;
    class A {
      int y = 0;
      A() {
        y = x + 1;
      }
    }
    A a = new A();
    // Line A
  }
}
```

Q1b: Fully qualify and draw stack and heap

```
class B {
  void f() {
    int x = 0;
    class A {
      int y = 0;
      A() {
        this.y = x + 1;
      }
    }
    A a = new A();
    // Line A
  }
}
```

Q1b: Fully qualify and draw stack and heap



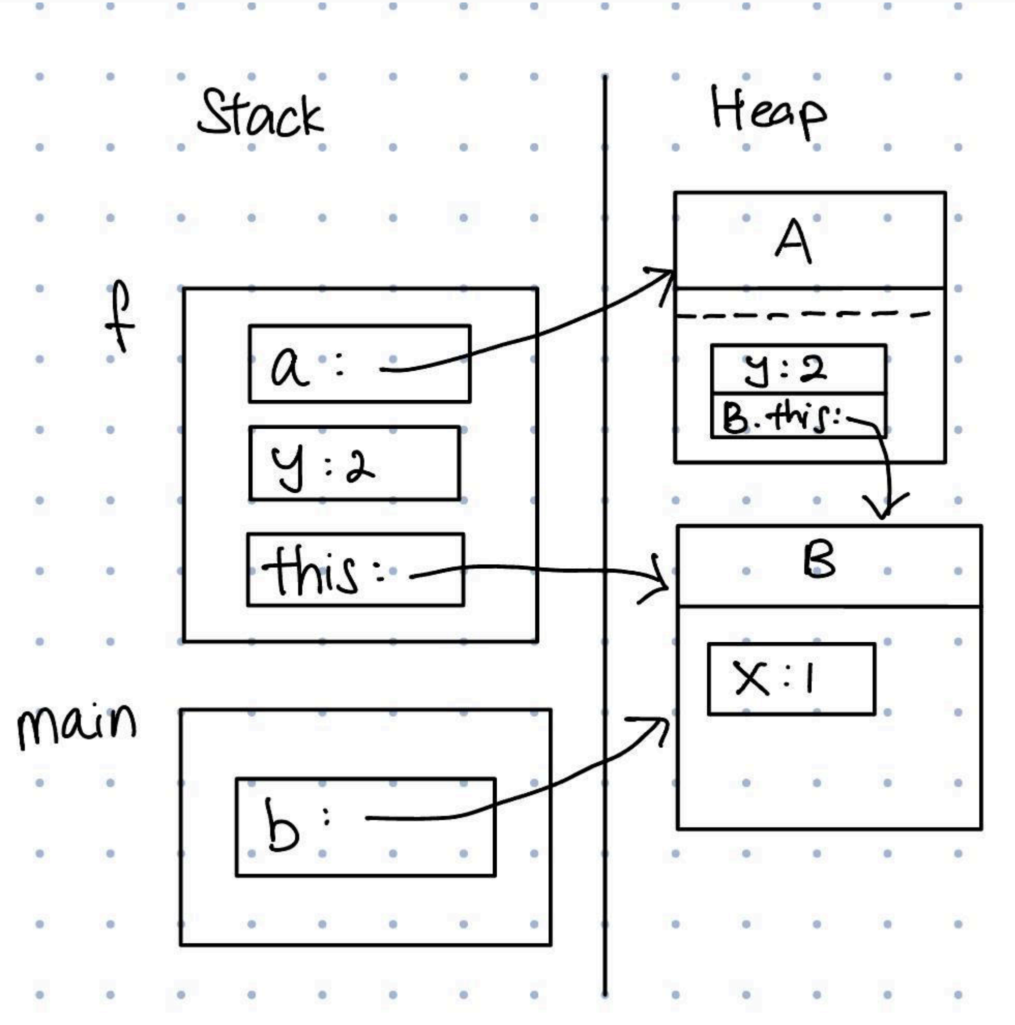
Q1c: Qualify and draw stack and heap

```
class B {
  int x = 1;
  void f() {
    int y = 2;
    class A {
      void g() {
        x = y;
      }
    }
    A a = new A();
    // Line A
    a.g();
  }
}
```

Q1c: Qualify and draw stack and heap

```java
class B {
  int x = 1;
  void f() {
    int y = 2;
    class A {
      void g() {
        B.this.x = y;
      }
    }
    A a = new A();
    // Line A
    a.g();
  }
}
```

Q1c: Qualify and draw stack and heap



Q2

- General steps to make a class immutable
  - Make class final
  - Make fields final
  - any mutator methods should create a new instance instead

The End

bye!