CS 61B Spring 2026 — Lists, Maps, Arrays, References

Discussion 02: January 26, 2026

1 Class Creation

Translate the following Python **Planet** class into Java. Make sure to use correct Java syntax and naming conventions (e.g. camelCase for methods). Assume **x**, **y**, and **mass** are doubles. For an example Java class, see https://tinyurl.com/dog-java.

```
import math

class Planet:
  def __init__(self, x, y, mass):
    self.x = x
    self.y = y
    self.mass = mass

  def distance_to(self, other):
    return math.sqrt(
      (other.x - self.x)**2 +
      (other.y - self.y)**2)

  @staticmethod
  def total_mass(planets):
    total = 0
    for p in planets:
      total += p.mass
    return total

p1 = Planet(5, 10, 100)
p2 = Planet(1, 2, 200)
p1.distance_to(p2)
Planet.total_mass([p1, p2])
```

2 Helpful LLM Use

On HW2, you had a chance to play around with large language models for solving the **starTriangle** problem. You've probably also used LLMs in prior programming classes.

What are some ways that you've used LLMs to help your learning? What are some ays that you've used LLMs that actually hindered your learning? Did you find it helpful to see what an LLM came up with for **starTriangle**?

3 List Exercises

(a) The code reference below shows the equivalent Java code for common `List` operations.

```java
List<String> lst = new ArrayList<>();
lst.add("zero");
lst.add("one");
lst.set(0, "zed");
System.out.println(lst.get(0));
System.out.println(lst.size());
if (lst.contains("one")) {
    System.out.println("one in lst");
}
for (String elem : lst) {
    System.out.println(elem);
}
```

```python
lst = []
lst.append("zero")
lst.append("one")
lst[0] = "zed"
print(lst[0])
print(len(lst))
if "one" in lst:
    print("one in lst")

for elem in lst:
    print(elem)
```

Fill in the method below which takes in two lists of integers and returns a new list containing the common items of the two given lists. Do not use the `retainAll` method.

```java
/** Returns a list containing the common items of the two given lists */
public static List<Integer> common(List<Integer> L1, List<Integer> L2) {
```

```python
rlist = []
for x in L1:
  if x in L2 and x not in rlist:
     rlist.append(x)

return rlist
```

(b) Fill in the method below which capitalizes all strings in the given list in place. Note that `"cat".toUpperCase()` returns `"CAT"`.

```java
/** Capitalizes all strings in the given list in place */
public static void capitalize(List<String> L) {
```

```python
# L is a list of strings
for i in range(len(L)):
  s = L[i]
  L[i] = s.upper()
```

4 Map Exercises

(a) The code reference below shows the equivalent Java code for common `Map` operations.

```java
Map<String, String> map = new HashMap<>();
map.put("hello", "hi");
map.put("hello", "goodbye");
System.out.println(map.get("hello"));
System.out.println(map.size());
if (map.containsKey("hello")) {
    System.out.println("\"hello\" in map");
}
for (String key : map.keySet()) {
    System.out.println(key);
}
```

```python
d = {}
d["hello"] = "hi"
d["hello"] = "goodbye"
print(d["hello"])
print(len(d))
if "hello" in d:
    print("\"hello\" in d")

for key in d.keys():
    print(key)
```

Translate the following Python function into Java. The function takes a list of integers and returns a map where the keys are the integers from the list, and the values are lists containing all integers from the original list that are strictly less than the key (without duplicates). Don't worry about order.

```java
/** Returns a map from each integer x in the list to a list (without duplicates)
  *  of all integers in the list that are less than x. */
 public static Map<Integer, List<Integer>> buildLessThanMap(List<Integer> L) {
```

```python
def build_less_than_map(L):
  result = {}
  for x in L:
    if x not in result:
      result[x] = []

    for y in L:
      if y < x:
        if y not in result[x]:
          result[x].append(y)
  return result

  ---

  example:
   L = [4, 1, 3, 3]
   m = build_less_than_map(L)
   m is: {1: [],
          3: [1],
          4: [1, 3]}
```

5 Positive Filter

Fill in the function below, which takes in a list of integers and returns a new array containing only the positive integers from the original list.

```
/** Returns an array containing only the positive integers from the given list */
public static int[] filterPositive(List<Integer> L) {









}
```

6 Particle References

What will the code below print?

```
public class Particle {
    public String flavor; public int lifespan;
    public Particle(String f, int l) {
        flavor = f;   lifespan = l;
    }
    public static void boil(Particle p) {
        p.flavor = "steam";
    }
    public static void decrement(int x) {
        x = x - 1;
    }
    public static void action(Map<Integer, Particle> m) {
        m.get(2).flavor = "lava";
        m.get(2).lifespan = 5;
    }
    public static void main() {
        Particle p1 = new Particle("water", -1);
        Particle p2 = new Particle("sand", -1);
        Map<Integer, Particle> m = Map.of(1, p1, 2, p2);
        boil(p1);
        IO.println(p1.flavor);
        decrement(p1.lifespan);
        IO.println(p1.lifespan);
        action(m);
        IO.println(p2.lifespan);
        IO.println(p2.flavor);
    }
}
```