Recitation 4

Lambda Expressions

---

1. For each of the following expressions, tell whether it is valid or not. If valid, explain the reasoning. If not valid, explain why.

    1. `() -> { }`

    2. `() -> "Hello"`

    3. `() -> { return "Goodbye"; }`

    4. `(Integer i) -> { return i+10; }`

    5. `(String s) -> { return "Bourne Ultimatum"; }`

---

2. Which of the following are functional interfaces?

    1. ```
       public interface Sum1 {
         int sum(int i, int j);
       }
       ```

    2. ```
       public interface Sum2 extends Sum1 {
         double sum(double i, double j);
       }
       ```

    3. ```
       public interface Rectangle {
         double getWidth();
         double getHeight();
         default double area() {
           return getWidth()*getHeight();
         }
       }
       ```

---

3. Write a sample lambda for each of the following:
    1. A boolean expression
    2. Creating an object
    3. Consuming from an object
    4. Select/extract from an object
    5. Combine two values
    6. Compare two objects

    For the questions involving objects, make up some class name - you don't have to write up the class.

---

4. Which of the following are valid uses of lambdas?

    1. ```
       public interface Executor {
         void execute();
       }
       public void do(Executor ex) {
         ex.execute();
       }
       do(() -> { });
       ```

2.
```
public interface Proc<T> {
   T process();
}
public Proc<String> get() {
   return () -> "I am a go getter!";
}
```

3.
```
Predicate<Student> p = (Student s) -> s.getMajor();
```

4.
```
BiFunction<Integer,Integer,String> bif = (int i, int j) -> ""+i+j;
```

---

5. This question refers to the Student class presented in lecture (see Sakai -> Resources -> Feb 16 -> Student.java)

    1. Write a NAMED lambda expression using a method reference to check if a student is a senior.
    2. Write a NAMED lambda expression using a method reference to get the major of a student.
    3. Given the following filter method:

```
public static List<T>
filter(List<T> list, Predicate<T> p) {
  List<T> res = new ArrayList<T>();
  for (T t: list) {
    if (p.test(t)) {
      res.add(student);
    }
  }
  return res;
}
```

    For each of the following, write one or more Predicate instances as NAMED lambda expressions that can be passed to the filter method to get the requires set of students. (Note: when composing predicates, you want to use named lambda expressions in the composition, otherwise the syntax gets unwieldy/unacceptable.)
        1. All non-CS majors
        2. All CS and Physics majors who are commuters
        3. Math seniors who are not commuters
        4. Resident non-Math non-freshman students