Questions

Bookmark this page

representations

1/1 point (graded)

Consider the following abstract data type.

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family, sorted from oldest to youngest, with no duplicates.
    public List<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return people;
    }
}
```

Here is a client of this abstract data type:

```
void client1(Family f) {
    // get youngest person in the family
    Person baby = f.people.get(f.people.size()-1);
    ...
}
```

Assume all this code works correctly (both Family and client1) and passes all its tests.

Now Family's representation is changed from a List to Set, as shown:

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family
    public Set<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return new ArrayList<>(people);
    }
}
```

Assume that Family compiles correctly after the change.

Which of the following statements are true about client1 after Family is changed? Check all that apply. You may find it useful to look at the online Java API documentation for List and Set to understand how client1 will be affected.

- [ ] client1 is independent of Family's representation, so it keeps working correctly.
- [x] client1 depends on Family's representation, and the dependency would be caught as a static error.
- [ ] client1 depends on Family's representation, and the dependency would be caught as a dynamic error.
- [ ] client1 depends on Family's representation, and the dependency would not be caught but would produce a wrong answer at runtime.
- [ ] client1 depends on Family's representation, and the dependency would not be caught but would (luckily) still produce the same answer.

Submit

Show Answer

Correct (1/1 point)

representations, part 2

1/1 point (graded)

Now consider client2:

```
void client2(Family f) {
    // get size of the family
    int familySize = f.people.size();
    ...
}
```

Which of the following statements are true about client2 after Family is changed? Check all that apply.

- [ ] client2 is independent of Family's representation, so it keeps working correctly.
- [ ] client2 depends on Family's representation, and the dependency would be caught as a static error.
- [ ] client2 depends on Family's representation, and the dependency would be caught as a dynamic error.
- [ ] client2 depends on Family's representation, and the dependency would not be caught but would produce a wrong answer at runtime.
- [x] client2 depends on Family's representation, and the dependency would not be caught but would (luckily) still produce the same answer.

✔

Submit

Show Answer

✔ Correct (1/1 point)

representations, part 3

1/1 point (graded)

Now consider client3:

```
void client3(Family f) {
    // get any person in the family
    Person anybody = f.getMembers().get(0);
    ...
}
```

Which of the following statements are true about client3 after Family is changed? Check all that apply.

- [x] client3 is independent of Family's representation, so it keeps working correctly.
- [ ] client3 depends on Family's representation, and the dependency would be caught as a static error.
- [ ] client3 depends on Family's representation, and the dependency would be caught as a dynamic error.
- [ ] client3 depends on Family's representation, and the dependency would not be caught but would produce a wrong answer at runtime.
- [ ] client3 depends on Family's representation, and the dependency would not be caught but would (luckily) still produce the same answer.

✔

Submit

Show Answer

✔ Correct (1/1 point)

representations, part 4

7/7 points (graded)

```
1  /**
2   * Represents a family that lives in a household together.
3   * A family always has at least one person in it.
4   * Families are mutable.
5   */
6  public class Family {
7      // the people in the family, sorted from oldest to youngest, with no duplicates.
8      private List<Person> people;
9
10     /**
11      * @return a list containing all the members of the family, with no duplicates.
12      */
13     public List<Person> getMembers() {
14         return people;
15     }
16 }
```

Which lines of code are part of the data type's specification, which are part of its representation, and which are part of its implementation?

lines 1-5

specification ✔

line 6

specification ✔

line 7

representation ✔

line 8

representation ✔

lines 10-12

specification ✔

line 13

specification ✔

line 14

implementation ✔

Submit

Show Answer

✔ Correct (7/7 points)

Previous Next