3. Relationships between Classes Results for Christine En-Tse Cheng

Correct answers are hidden.

Submitted Dec 13 at 5:21p.m.

Unanswered

Question 1

0 / 1 pts

Which of the following code fragments do not result in an error?

```
public class Parent {
  public int a;

  public Parent(int number) {
    this.a = number;
  }
}

public class Child extends Parent {
  public Child() {}
}
```

```
public class Parent {
  public int a;

  public Parent(int number) {
    this.a = number;
  }
}

public class Child extends Parent {
  public Child(int numb) {
  }
}
```

```
public class Parent {}
public class Child extends Parent {}
```

```
public class Parent {
  public int a;

  public Parent() {
    this.a = 0;
  }
}

public class Child extends Parent {
  public Child() {}
}
```

```
public class Parent {
  public int a;

  public Parent(int number) {
    this.a = number;
  }
}

public class Child extends Parent {
  public Child(int numb) {
    super(numb);
  }
}
```

```
public class Parent {
  public int a;

  public Parent() {}

  public Parent(int number) {
    this.a = number;
  }
}

public class Child extends Parent {
  public Child(int numb1, int numb2) {
  }
}
```

```
public class Parent {
  public int a;

  public Parent() {}

  public Parent(int number) {
    this.a = number;
  }
}

public class Child extends Parent {
  private int b;

  public Child(int numb) {
    super();
    this.b = numb;
  }
}
```

Unanswered

Question 2

0 / 1 pts

Consider this code:

```
public class Parent {

  public void func() {
    System.out.println("parent");
  }
}
```

Which of the following code fragments will print "parent"?

- [ ]
```
public class Child extends Parent {

  public void func() {
    super.func();
  }

  public static void main(String[] args) {
    new Child().func();
  }
}
```

- [ ]
```
public class Child extends Parent {

  public static void main(String[] args) {
    new Child().func();
  }
}
```

- [ ]
```
public class Child extends Parent {
  private char c = 'c';

  public void func() {
    super();
  }

  public static void main(String[] args) {
    new Child().func();
  }
}
```

Unanswered

Question 3

0 / 1 pts

From the statements below, select the ones that are true.

- [ ] Interfaces can declare instance variables.
- [ ] Interfaces cannot have implemented methods.
- [ ] Interfaces can declare method signatures.
- [ ] A class can implement multiple interfaces
- [ ] A class can extend multiple other classes
- [ ] Abstract classes cannot have implemented methods.

Unanswered

Question 4

0 / 1 pts

Suppose we have the following structure:

- `Vegetable` is an abstract class
- `Shreddable` is an interface
- `Carrot` extends `Vegetable` and implements `Shreddable`
- `Pumpkin` extends `Vegetable` and has an extra method called `squash`
- Neither `Carrot` nor `Pumpkin` are abstract classes

Suppose we have the following variables defined that refer to non-null objects:

`Carrot c1 = new Carrot(...)`

`Pumpkin p1 = new Pumpkin(...)`

`Vegetable v1 = ... // either new Carrot(...)` or `new Pumpkin(...)`

Select the code fragments that definitely do not result in any error. Consider each code fragment independently.

- [ ] Vegetable[] vegetables = {p1, v1, c1};
- [ ] Shreddable s = v1;

- [ ] Pumpkin p = c1;
- [ ] Vegetable v1 = (Vegetable) c1;
- [ ] Vegetable v = p1;
- [ ] Shreddable s = (Shreddable)v1;
- [ ] ((Pumpkin) v1).squash();
- [ ] Carrot c = v1;
- [ ]
```
Vegetable v = p1;
v.squash();
```
- [ ] Shreddable s = c1;
- [ ] Vegetable v1 = (Carrot) c1;

Unanswered

Question 5

0 / 1 pts

Recall the class structure from the previous question:

- `Vegetable` is an abstract class
- `Shreddable` is an interface
- `Carrot` extends `Vegetable` and implements `Shreddable`
- `Pumpkin` extends `Vegetable` and has an extra method called `squash`
- Neither `Carrot` nor `Pumpkin` are abstract classes

Suppose we have the following code:

```
Carrot c1 = new Carrot(...);
Vegetable v1 = c1;
```

Which of the following expressions evaluate to `true`?

- [ ] c1 instanceof Carrot
- [ ] v1 instanceof Carrot
- [ ] v1 instanceof Pumpkin
- [ ] c1 instanceof Shreddable
- [ ] c1 instanceof Vegetable
- [ ] c1 == v1
- [ ] v1 instanceof Shreddable
- [ ] v1 instanceof Vegetable

Unanswered

Question 6

0 / 1 pts

In Java, overriding means ...

- ○ that a subclass has a method with the same signature as a method in its super class.
- ○ to override an existing variable.
- ○ that a class has two or more methods with the same name but with different parameters.

Unanswered

Question 7

0 / 1 pts

Which of the following code fragments do not result in an error?

- [ ]
```
interface Dog {
  void eat(String);
}
```
- [ ]
```
interface Dog {
  private int age = 2;
```

```
}
```

```
interface Dog {
  default void bark() {
    System.out.println("Bow-wow");
  }
}
```

```
interface Dog {
  String species;
}
```

```
interface Dog {
  static void bark();
}
```

```
interface Dog {
  String SPECIES = "mammal";
}
```

```
interface Dog {
  void bark() {
    System.out.println("Bow-wow");
  }
}
```

```
interface Dog {
  void bark();
}
```

```
interface Dog {
}
```

```
interface Dog {
  void eat(String food);
}
```