What are some good java interview questions and answers regarding generics and annotations? [closed]

Ask Question

What are some good java interview questions and answers regarding generics and annotations?

java generics annotations

edited May 21 '10 at 15:46
David M
3,232 1 14 31

asked May 21 '10 at 15:15
Daniel Honig
1,955 5 18 24

closed as off-topic by Dukeling, Chris, gnat, Tony Hopkinson, nsfyn55 May 31 '14 at 19:31

- This question does not appear to be about programming within the scope defined in the help center.

If this question can be reworded to fit the rules in the help center, please edit the question.

4 Poll questions should be marked community wiki. – danben May 21 '10 at 15:21

What level of developer are you looking for? Lead, or competent and midlevel? – Dean J May 21 '10 at 15:33

1 This question appears to be off-topic because it is about interview questions – Tony Hopkinson May 31 '14 at 19:07

11 Answers

Since Java 5 came out, I have seen dozens of people not understand why, given an interface `I`, and classes `A` and `B extends A` you can't pass an `I<B>` where an `I<A>` is required. A lot of people find it counter-intuitive.

To test a person's ability to *reason* about Generics, then I would first ask them if it *is* possible to assign an `I<B>` to an `I<A>` reference as described above. If not, why not? If they get it wrong, tell them they're wrong and ask them to try to fill in the blanks here to show why this example would be type un-safe if it could compile:

```
   //...
   List<String> list = new LinkedList<String>();
   someMethod(list);
   //blank 1
}
public void someMethod(List<Object> list) {
   //blank 2
}
```

At this point it should be pretty easy and I would be a bit worried if they couldn't construct such an example. An example is

```
//blank 1
String item = list.get(0);

//blank 2
list.add(Integer.valueOf(5));
```

answered May 21 '10 at 15:40

Mark Peters
64.4k 11 133 171

Now I got it,Thanks – Gagandeep Singh Jan 21 '16 at 17:48

This test:

http://tests4geeks.com/test/java

contains some questions about annotations.

It does not contain any questions about generic. But instead of it, there are some other interesting themes like:

Multi-Threading,

Memory,

Algorithms and Data Structures,

OOP,

etc.

answered Mar 14 '13 at 11:48
Vadim
131 2 2

I prefer your solution.. – Guru Mar 15 '13 at 10:11

some of the questions there are written by non native english speakers (or all the questions). for most does not matter but for a few it garbles the question. – tgkprog Jul 21 '13 at 21:58

- What does the `Class Enum<E extends Enum<E>>` ensure?
- Is `<T> List<? extends T> x()` a useful signature?
- Annotation retentions policies and why they are in place?
- Suppose you would like to reuse a class in different contexts would you use annotations or external configuration? (i.e. annotation introduce dependencies).

Harder:

- Examples of a valid generic type that cannot be expressed with the Java type system and will lead to compiler warnings.
- Examples where the java compiler will/will not infere the generic type? (examples of code where the compiler will infere the wrong type)
- What generic type information is not erased and can be retrieved from the byte code? (Super type tokes are one application)

- What's APT (use cases when not to use reflection)?

edited May 21 '10 at 16:11

answered May 21 '10 at 15:29
Thomas Jung
25.4k 7 68 105

Thomas, could you please elaborate (or provide a link) for generic type information which is not erased (bullet 3 of harder questions)? It's very interesting... – Oak May 21 '10 at 15:57

1 @Oak - Added it. – Thomas Jung May 21 '10 at 16:13

Question: How can you determine what type of object a generic is using at runtime?

Answer: It is not possible to determine the type because of type erasure.

answered May 21 '10 at 15:23
dbyrne
36.1k 10 75 100

1 Not exactly true. Good old super type tokens can only be implement because some type information is left. There is another exception to the nice but wrong rule, but I've sadly forgot it. – Thomas Jung May 21 '10 at 15:37

1 Well, the type of an object at runtime is obtained by calling `object.getClass()` , irrespective of anything using it, so you might want a more carefully worded question. – Pete Kirkham May 21 '10 at 16:07

---

1. Generic:

Q: What is the difference between a Hashmap and a Hashtable?

A: Hashtable is synchronized, Hashmap is not.

2. Annotations:

Q: Describe serializing java objects with the javax.xml.bind.Marshaller interface and annotations.

A: Describing something like this in a meaningful context ought to be acceptable:

```
@XmlRootElement

class Employee {

...

}
```

answered May 21 '10 at 15:45
LoudNPossiblyWrong
2,015 4 27 42

6 I think you're confusing Generics with the Java Collections library. – Mark Peters May 21 '10 at 15:56

Generics: Ask a question designed to see if they understand type erasure.

Annotations: Ask them what their favorite annotation is, and how it works (you don't need a detailed technical explanation, but you're looking for something more than "magic").

answered May 21 '10 at 15:20
Hank Gay
50.7k 23 134 207

A nice type erasure question is which signatures are not possible due to type erasure (e.g. x(List<Object>) and x(List<String>) together). – Thomas Jung May 21 '10 at 15:26

Annotations: What are the risks? (the compiler can get into an infinite loop and jam the build process).

Generics: How to build a mixin using generics? (write a generic class that takes a parameter, and then extend it with the sub-class as the parameter).

Also, +1 on type erasure.

answered May 21 '10 at 15:21

Little Bobby Tables
3,639 2 22 35

1 Reference on "compiler can get into an infinite loop" for annotations as a risk. I thought self-referential annotations were explicitly prevented for this reason. – Nate May 21 '10 at 16:59

@Nate - I too, am curious to see how this could happen. – Tom Tresansky May 21 '10 at 19:17

In one of two ways: 1. The annotation processing code gets stuck, for no reason - E.g., imagine an annotation that calculates a factorial and gets @Fact(-1). 2. An annotation processing code that generates code, marks the new code for another round of compilation, and the new code also includes annotations... I don't remember the exact flag for that. – Little Bobby Tables May 22 '10 at 18:09

If you are looking for a rock star Java programmer, you could create quite a few advanced questions from the Generics chapter in Bloch's Effective Java. Things like the homogeneous or heterogenous containers (the bird cage and lion cage examples from one of the java tutorials), erasure, etc.

In less senior folks I primarily look for an understanding of why one would want generics (you'd be surprised how many folks don't appreciate that and believe the Java 2 way of doing things still rules), and things like that.

In annotations, I like asking about the "Override" annotation and its benefits. I also like having a deeper discussion about the pros and cons of annotations and when it is appropriate to use them - I'm not a fan of overzealous meta programming. It's also a good chance to see if someone used Hibernate or JUnit with annotations.

answered May 21 '10 at 15:26
Uri
65.7k 42 196 300

Effective Java is an essential book. Will have a look into that chapter. – Daniel Honig May 21 '10 at 15:36

Here are a few I just made up:

-- [Cagey generics] Would uncommenting any of these lines cause problems? Which ones, if any? Why or why not?

```
public class Cage<T> { ... }
public class Animal { ... }
public class Bear extends Animal { ... }
// Cage<Animal> c = new Cage<Bear>();
// Cage<Bear> c = new Cage<Animal>();
// Cage<?> c = new Cage<Animal>();
// Cage<Animal> c = new Cage<?>();
```

-- [Constraints] Only `Animals` should go into `Cages` . How can we tighten up the class definitions above so that they reflect this new requirement?

-- [Legal troubles] You cannot instantiate an array of a generic type. Thus, something like `new List<Animal>[10]` is illegal. Can you think of a scenario where, if this *were* legal, you would run into trouble?

Answers left as an exercise to the reader -- you're not going to learn anything if you don't figure them out yourself! But here are some hints.

- [Cagey generics]: What does `?` mean? Do you remember the term "covariance"?
- [Constraints]: Java lets you constrain the values of a type parameter. Do you remember the syntax for this?
- [Legal troubles]: Suppose you could do `Object[] arr = new List<String>[]` . Can you write some code that puts something into `arr` 's `Lists` that shouldn't go in there?

answered May 21 '10 at 15:31

John Feminella

222k 32 295 322

"What type of things are annotations good at?" and "what type of things are annotations bad at?" comes to mind.

Annotations are good at meta-programming, but as a possible best-practice, code that worked *with* the annotations in should still work if you take all of them out.

Or not. Your mileage may vary, but you probably want all of your senior developers to agree on that one.

answered May 21 '10 at 15:34

Dean J

20.3k 13 52 90

Here are a couple on generics:

1)

Question: If I had this method, how would I create a new instance of type `T` and assign it to a variable called `item` inside this method (assume `Class<T>` has a default constructor)?

```
public static <T> Collection<T> select(Class<T> c, String sqlStatement) {
}
```

Answer:

```
T item = c.newInstance();
```

2)

Question: If you wanted to indicate that the generic extends some base class, but you don't know what the class that extends the base class is going to be until runtime, how could you declare the generic?

Answer: Use a wildcard: `<? extends SomeBaseClass>`

edited May 21 '10 at 16:57

answered May 21 '10 at 15:22

dcp
40.9k 16 114 143

1.) assumes default constructor. – Nate May 21 '10 at 16:45

@Nate - Fair point, I updated it to reflect this. – dcp May 21 '10 at 16:57