David S. Anderson

Md Ashiqur Rahman

CSc 210 – Summer 2020

23 July 2020

PA7 – Written Portion

1. *What is the purpose of generics? Why do we need them?*

Generics were introduced into Java for many reasons. One of the most important reasons is that generics allow programmers to reuse the same code with a variety of different inputs, which can save massive amounts of time when a project is large and multi-layered. Generics also cause Java to implement type-checking at compile time rather than at run-time, which prevents some bugs that might otherwise have made it through testing. Furthermore, generics make it so programmers don't have to do casting, because the type has already been specified and checked by the Java compiler ahead of time. Without this feature we would be unable to use the myriad tools that have been added to the Java toolkit over the years that require us to specify a type.

2. *Why don't we just use Object as the parameterized type for our generic classes? Then we could put absolutely any object into our list.*

We don't just use Object as the parameterized type for our generic classes because that would essentially defeat the purpose. That would be similar to providing no type at all, since all classes are children of Java's Object superclass. This would cause errors at run-time because programmers could store multiple types in the same list, but when those items were called Java

would not know the type and therefore could not store them and use them properly. Generic types allow us to reuse the code by supplying a new type each time we need one.

3. *In class we made MyLinkedList generic and not MyArrayList. Why did we choose to make MyLinkedList generic first? Are there any issues when combining generics and arrays?*

We chose to make MyLinkedList generic because arrays and generics are incompatible and run into issues. The primary reason for this is that arrays are covariant (we can assign a subclass type array to its superclass array reference, like initializing it with Object type but then indicating Integer) while generics are invariant (we are unable to assign subclass type to its superclass reference). One good example would be trying to initialize an ArrayList of integers with a fixed value.

4. *How many type parameters are you allowed to use in a generic class?*

I don't think there is a limit on the number of type parameters one could use in a generic class. In lecture we learned the various naming conventions for generic type parameters up to the 4th parameter, but I believe one could theoretically use as many as necessary. However, I don't think this would often be an issue.

5. *Explain Type Erasure. How does it work?*

Type erasure is the process Java uses during compile-time to ensure that no errors are thrown and to replace the parameterized type with the Object type (or the supplied bound but we didn't go in-depth about that concept). Java then inserts type casts to preserve type safety and generates

bridge methods to keep polymorphism in extended generic types (*TutorialsPoint.com*). This is why we are able to reuse the same code and supply different type parameters repeatedly for our generic classes.