CS2030S Cheatsheet

for midterms AY23-24, Sem 2

m. **zaidan**

Types

Subtypes $S <: T$ or $T >: S$ (S is subtype of T)
1. **Reflexive** $S <: S$
2. **Transitive** $S <: T \land T <: U \implies S <: U$
3. **Antisymmetry** $S <: T \land T <: S \implies S = T$

Subtyping Relationship
byte <: short <: int <: long <: float <: double
char <: int

Widening and Narrowing Type Conversion
Widening Variable of type T can hold a value of type S if $S <: T$
Narrowing Variable of type S cannot hold a value of type T if $S <: T$

Variance of Types

Covariance $S <: T \implies C(S) <: C(T)$
Contravariance $S <: T \implies C(T) <: C(S)$
Invariant Neither

OOP Principles

Encapsulation

Information Hiding Private instance fields

Abstraction

Do not show actual implementation of methods.

Inheritance

Ability to reuse code of existing super classes.
Models **is-a** relationship.

Polymorphism

Using same method signatures in subclasses to determine behaviour for specific subclasses.
–
Tell Don't Ask The client should not be doing computation on the object's behalf.

Method Signature and Descriptor

Method Signature method name, number of parameters, type of each parameter and order of parameters
Method Descriptor method signature + return type

Liskov Substitution Principle

Let $\phi(x)$ be a property provable about objects x of type T.
Then $\phi(y)$ should be true for objects y of type S where $S <: T$

Interfaces and Abstract Classes

Interface	**Abstract**
Can implement multiple	Can only extend one
Only abstract methods	Abstract and non-abstract

Dynamic Binding

1. Determine compile-time type of target
2. Check all accessible methods (including inherited ones)
3. Most specific one callable
4. Determine run-time type of target
5. Determine method called.

Wildcards

PECS Producer Extends, Consumer Super

Upper-Bounded Wildcards A < ? extends S >
Covariant.

Lower-Bounded Wildcards A < ? super S >
Contravariant.

Raw Type

A<?> Complex type of a specific but unknown type.
A<Object> Complex type of Object instances with type checking.
A Complex type of Object instances without type checking.

Type Inference

Constraints
1. Target Type
2. Argument Type
3. Type Parameter Bound