

- #1 Code reuse that allows a programmer to use those portions of the original code while modifying other portions to fit their specific needs describes
- (a) polymorphism.
 - (b) automorphism.
 - (c) objectivism.
 - (d) inheritance.
 - (e) code duality.
- #2 In an object oriented language, an abstract data type is generally known as a(n)
- (a) object.
 - (b) record.
 - (c) instance.
 - (d) class.
 - (e) struct.
- #3 The calls to an object's subprograms are sometimes referred to as _____ to that object.
- (a) beacons
 - (b) messages
 - (c) recitations
 - (d) invocations
 - (e) requests
- #4 Dynamic binding of subprogram calls to subprogram definitions is needed in order to support
- (a) polymorphism.
 - (b) platform independence.
 - (c) encapsulation.
 - (d) envelopment.
 - (e) inheritance.
- #5 The design issue that centers around whether simple data types are treated as objects is termed the
- (a) wrapper class issue.
 - (b) multiple inheritance issue.
 - (c) primitive typing issue.
 - (d) exclusivity of objects issue.
 - (e) subtype issue.

Enter the letter(s) of each answer below. You may choose multiple answers, but credit will be divided by the number of choices made.

1 D 2 D 3 B 4 A 5 D 6 A 7 B 8 E 9 A 10 B

CS-3160 Multiple Choice Questions

- #6 A common way to mitigate the disadvantages associated with multiple inheritance is to use
- (a) interfaces.
 - (b) overridden subprogram.
 - (c) diamond inheritance.
 - (d) graphic inheritance.
 - (e) parametric polymorphism.
- #7 The motivation for allowing static binding of class subprogram calls is to improve
- (a) security.
 - (b) performance.
 - (c) reliability.
 - (d) maintainability.
 - (e) writability.
- #8 Which levels of concurrency pose challenges and issues for programming language design?
- (a) program and subprogram
 - (b) instruction and unit
 - (c) instruction and statement
 - (d) instruction and program
 - (e) statement and subprogram
- #9 What distinguishes a heavyweight task from a lightweight task?
- (a) Heavyweight tasks run in their own address space while lightweight tasks do not.
 - (b) Heavyweight tasks account for more than half of processor execution time while lightweight tasks do not.
 - (c) Heavyweight tasks require memory access while lightweight tasks only use processor registers.
 - (d) Heavyweight tasks are active throughout program execution while lightweight tasks are only active when needed.
 - (e) Heavyweight tasks are callable from anywhere in the program while lightweight tasks must be called from object instances.
- #10 The two types of synchronization that are required when tasks that share resources are
- (a) one-way and multi-way.
 - (b) cooperation and competition.
 - (c) synchronous and asynchronous.
 - (d) joint and disjoint.
 - (e) cooperation and asynchronous.