Top of Form

|  |  |
| --- | --- |
| **1.** | which data type is narrower among? |
| |  | | --- | | A.  short |  |  | | --- | | B.  float |  |  | | --- | | C.  byte | | | |
|  | | |
| **2.** | which data type is narrower among? |
| |  | | --- | | A.  long |  |  | | --- | | B.  float |  |  | | --- | | C.  double | | | |
|  | | |

|  |  |
| --- | --- |
| **3.** | which one is automatic |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |
| **4.** | short assigning to int is |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |

|  |  |
| --- | --- |
| **5.** | long assigning to double is |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |
| **6.** | class A  {  public static void main(String[] args)  {  byte b = 100;  short s = 1000;  s = b;  System.out.println("done");  }  } |
| |  | | --- | | A.  Compilation error |  |  | | --- | | B.  done | | | |
|  | | |

|  |  |
| --- | --- |
| **7.** | class B  {  public static void main(String[] args)  {  byte b = 100;  short s = 1000;  s = (short) b;  System.out.println("done");  }  } |
| |  | | --- | | A.  Compilation error |  |  | | --- | | B.  done | | | |
|  | | |
| **8.** | class C  {  public static void main(String[] args)  {  byte b = 100;  short s = 1000;  int i;  double d;  s = b;  i = b;  i = s;  d = b;  d = s;  d = i;  System.out.println("done");  }  } |
| |  | | --- | | A.  Compilation error |  |  | | --- | | B.  done | | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **9.** | class D  {  public static void main(String[] args)  {  System.out.println("main begin");  byte b = 20;  test(b);  System.out.println("main end");  }  public static void test(int i)  {  System.out.println("test");  }  } | |
| |  | | --- | | A.  Compilation error |  |  | | --- | | B.  done |  |  | | --- | | C.  main begin  test  main end | | | | |
|  | | | |
| **10.** | | class E  {  public static void main(String[] args)  {  System.out.println("main begin");  byte b = 20;  test(b);  short s = 20;  test(s);  int i = 20;  test(i);  long j = 20;  test(j);  System.out.println("main end");  }  public static void test(double i)  {  System.out.println("test");  }  } |
| |  | | --- | | A.  Compilation error |  |  | | --- | | B.  main begin  test  test  test  test  main end |  |  | | --- | | C.  main begin  test  main end | | | | |
|  | | | |

|  |  |
| --- | --- |
| **11.** | which data type is wider among? |
| |  | | --- | | A.  short |  |  | | --- | | B.  float |  |  | | --- | | C.  byte | | | |
|  | | |
| **12.** | which data type is wider among? |
| |  | | --- | | A.  long |  |  | | --- | | B.  float |  |  | | --- | | C.  double | | | |
|  | | |

|  |  |
| --- | --- |
| **13.** | which one an explicit |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |
| **14.** | long assigning to int is |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |

|  |  |
| --- | --- |
| **15.** | double assigning to byte is |
| |  | | --- | | A.  widening |  |  | | --- | | B.  narrowing | | | |
|  | | |
| **16.** | which is widermost |
| |  | | --- | | A.  long |  |  | | --- | | B.  float |  |  | | --- | | C.  double | | | |
|  | | |

|  |  |
| --- | --- |
| **17.** | which is narrowest |
| |  | | --- | | A.  long |  |  | | --- | | B.  byte |  |  | | --- | | C.  double | | | |
|  | | |
| **18.** | class I  {  public static void main(String[] args)  {  int x = 1000;  byte y = 10;  y = x;  System.out.println("done");  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  done | | | |
|  | | |

|  |  |
| --- | --- |
| **19.** | class I  {  public static void main(String[] args)  {  int x = 1000;  byte y = 10;  y = x;  System.out.println("done");  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  done | | | |
|  | | |
| **20.** | class J  {  public static void main(String[] args)  {  int x = 1000;  byte y = 10;  y = (byte) x;  System.out.println("done");  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  done | | | |
|  | | |

|  |  |
| --- | --- |
| **21.** | class K  {  public static void main(String[] args)  {  int x = 129;  byte y = 10;  y = (byte) x;  System.out.println(x + "," + y);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  129,-127 |  |  | | --- | | C.  129,129 | | | |
|  | | |
| **22.** | class L  {  public static void main(String[] args)  {  int x = -130;  byte y = 10;  y = (byte) x;  System.out.println(x + "," + y);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  129,-127 |  |  | | --- | | C.  -130,126 | | | |
|  | | |

|  |  |
| --- | --- |
| **23.** | class M  {  public static void main(String[] args)  {  int i = 10;  short j = i;  System.out.println(j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  10 | | | |
|  | | |
| **24.** | class N  {  public static void main(String[] args)  {  int i = 32768;  short j = (short) i;  System.out.println(j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  32768 |  |  | | --- | | C.  -32768 | | | |
|  | | |

|  |  |
| --- | --- |
| **25.** | class O  {  public static void main(String[] args)  {  int i = 130;  short j = (byte) i;  System.out.println(j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  -126 |  |  | | --- | | C.  -130 | | | |
|  | | |
| **26.** | incase of derive data type, upcasting is an automatic. |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |

|  |  |
| --- | --- |
| **27.** | incase of derive data type, upcasting should be an explicit. |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |
| **28.** | incase of derive data type, downcasting is an automatic. |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |

|  |  |
| --- | --- |
| **29.** | incase of derive data type, upcasting should be an explicit. |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | | |
|  | | | |
| **30.** | method argument is super class type and while calling the same method supplying subclasstype is proper syntax. | |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | | |
|  | | | |

|  |  |
| --- | --- |
| **31.** | method argument is subclass type and while calling the same method supplying super classtype is proper syntax. |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | | |
|  | | | |
| **32.** | method return type is Object type and while calling the same, is it possible to take the return value in to String type variable? | |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | | |
|  | | | |

|  |  |
| --- | --- |
| **33.** | method return type is String type and while calling the same, is it possible to take the return value in to Object type variable? |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |
| **34.** | method return type is String type and while calling the same, is it possible to take the return value in to Object type variable? |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |

|  |  |
| --- | --- |
| **35.** | Object obj = new String(); //will it compile |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |
| **36.** | String obj = new Object(); //will it compile |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |

|  |  |
| --- | --- |
| **37.** | Object obj = (Object) new String(); //will it compile |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |
| **38.** | String obj = (String) new Object(); //will it compile |
| |  | | --- | | A.  yes |  |  | | --- | | B.  no | | | |
|  | | |

|  |  |
| --- | --- |
| **39.** | Which option gives runtime excfeption? |
| |  | | --- | | A.  Object obj = new Object() |  |  | | --- | | B.  Object obj = new String() |  |  | | --- | | C.  String obj = (String) new Object(); |  |  | | --- | | D.  Object Obj = (Object) new String(); | | | |
|  | | |
| **40.** | What is the output?  String s1 = new String();  System.out.print(s1 instanceof Object);  System.out.print(s1 instanceof String); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |

|  |  |
| --- | --- |
| **41.** | What is the output?  Object s1 = new String();  System.out.print(s1 instanceof Object);  System.out.print(s1 instanceof String); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |
| **42.** | What is the output?  String s1 = new Object();  System.out.print(s1 instanceof Object);  System.out.print(s1 instanceof String); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |

|  |  |
| --- | --- |
| **43.** | What is the output?  String s1 = (String) new Object();  System.out.print(s1 instanceof Object);  System.out.print(s1 instanceof String); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |
| **44.** | What is the output?  Object s1 = new Object();  System.out.print(s1 instanceof Object);  System.out.print(s1 instanceof String); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |

|  |  |
| --- | --- |
| **45.** | What is the output?  String s1 = new String("4.5");  System.out.print(s1 instanceof Integer);  System.out.print(s1 instanceof Double); |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  truetrue |  |  | | --- | | D.  truefalse |  |  | | --- | | E.  falsefalse | | | |
|  | | |
| **46.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  A a1 = new A();  System.out.print(a1.i);  System.out.print(a1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |

|  |  |
| --- | --- |
| **47.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  A a1 = (B)new A();  System.out.print(a1.i);  System.out.print(a1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |
| **48.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  A a1 = new B();  System.out.print(a1.i);  System.out.print(a1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |

|  |  |
| --- | --- |
| **49.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  A a1 = (B) new B();  System.out.print(a1.i);  System.out.print(a1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |
| **50.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  B b1 = new B();  System.out.print(b1.i);  System.out.print(b1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |

|  |  |
| --- | --- |
| **51.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  B b1 = new A();  System.out.print(b1.i);  System.out.print(b1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |
| **52.** | What is the output?  class A  {  int i = 10;  }  class B extends A  {  int j = 20;  }  class C  {  public static void main(String[] args)  {  B b1 = (B) new A();  System.out.print(b1.i);  System.out.print(b1.j);  }  } |
| |  | | --- | | A.  compilation error |  |  | | --- | | B.  runtime error |  |  | | --- | | C.  1020 |  |  | | --- | | D.  2010 |  |  | | --- | | E.  1010 | | | |
|  | | |

|  |  |
| --- | --- |
| **53.** | Which of the following is a derived data type |
| |  | | --- | | A.  class |  |  | | --- | | B.  interface |  |  | | --- | | C.  int |  |  | | --- | | D.  Both a and b |  |  | | --- | | E.  Both b and c | | | |
|  | | |
| **54.** | The members which are under one inheritance are involved in derived casting |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |

|  |  |
| --- | --- |
| **55.** | Is it possible to use use sub class where ever supper class is required |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |
| **56.** | Which type of casting is automatic |
| |  | | --- | | A.  Down casting |  |  | | --- | | B.  Up casting | | | |
|  | | |

|  |  |
| --- | --- |
| **57.** | Object class is the super most class |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |
| **58.** | We need to do up casting explicitly |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |

|  |  |
| --- | --- |
| **59.** | Compiler only do the down casting automatically |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |
| **60.** | Every class is a sub class to object classes |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |

|  |  |
| --- | --- |
| **61.** | Is it possible to inject variables once after object got created |
| |  | | --- | | A.  Yes |  |  | | --- | | B.  No | | | |
|  | | |
| **62.** | To avoid getting class cast exception we can use instanceof operator |
| |  | | --- | | A.  Yes |  |  | | --- | | B.  No | | | |
|  | | |

|  |  |
| --- | --- |
| **63.** | If object type is a specific class type instanceof operator returns |
| |  | | --- | | A.  True |  |  | | --- | | B.  False | | | |
|  | | |
| **64.** | Is it possible to use instanceof accross the classes |
| |  | | --- | | A.  Yes |  |  | | --- | | B.  No | | | |
|  | | |

Bottom of Form